Linearity and Bisimulation

Nobuko Yoshida* Kohei Honda† Martin Berger†

Abstract. Exploiting linear type structure, we introduce a new theory of weak bisimilarity for the π-calculus in which we abstract away not only τ-actions but also non-τ actions which do not affect well-typed observers. This gives a congruence far larger than the standard bisimilarity while retaining semantic soundness. The framework is smoothly extendible to other settings involving nondeterminism and state. As an application we develop a behavioural theory of secrecy in the π-calculus which ensures secure information flow for a strictly greater set of processes than the type-based approach in [20, 23], while still offering compositional verification techniques.

1 Introduction

Linearity is a fundamental concept in semantics with many applications to both sequential and concurrent computation. This paper studies how a linear type structure, close to those of Linear Logic [12] and game semantics [6, 22, 24], can be used to give a powerful extension of a basic process equivalence, bisimilarity. We use a linear π-calculus introduced in [35], though the framework is adaptable to various type structures which combine linearity with other elements such as state and nondeterminism. A central idea is that observables, an underpinning of any behavioural semantics, can be given a radical change by exploiting the linear type structure. The resulting bisimilarity is strictly larger than the standard construction while retaining semantic soundness. As an application we develop a behavioural theory of secrecy which, via semantic means, ensures secrecy for a strictly larger set of processes than the type-based approach in [20, 23].

Let us briefly explain the key ideas of the new bisimilarity, using a process encoding of a λ-calculus. We first recall that the linear π-calculus in [35] can fully abstractly embed $\lambda_{()\times+}$, the simply typed λ-calculus with unit, products and sums. The encoding $[\![M : \alpha]\!]_u$ for a λ-term $M : \alpha$ in [35] is a typed version of Milner's encoding [26]. We also recall that in $\lambda_{()\times+}$, the following equation is semantically sound: $\Gamma \vdash M_1 = M_2 : \mathtt{unit}$ for any $\Gamma \vdash M_{1,2} : \mathtt{unit}$. Categorically this comes from the uniqueness of arrows from each object to the final one. Operationally this is justified because the closed terms of this type are always β-equal to its unique constant, which we write $\star$. For example we have the following equation:

$$y : \mathtt{unit} \Rightarrow \mathtt{unit} \vdash \ (y\star) = \star : \mathtt{unit}$$

If we apply the encoding in [35] to this, we obtain the following two processes:

$$[\![(y\star)]\!]_u \stackrel{\text{def}}{=} !u(c).\overline{y}(e)e.\overline{c} \qquad\qquad [\![\star]\!]_u \stackrel{\text{def}}{=} !u(c).\overline{c}.$$

* University of Leicester, U.K. † Queen Mary, University of London, U.K.

Here $x(y)$ is an input of y via x, $\overline{x}(y)$ is an (asynchronous) output of a fresh name y via x, and ! indicates replication. Thus, the first process, $[\![(y\star)]\!]_u$, when invoked at u with a continuation c, first asks at y and, after receiving an answer at e, returns to c; while $[\![\star]\!]_u$ immediately answers at the continuation after the invocation. Because of the obvious difference in these actions, we know $[\![(y\star)]\!]_u \not\approx [\![\star]\!]_u$ where $\approx$ is the standard weak bisimilarity. However, since the encoding is fully abstract, the contextual equivalence $\cong_\pi$ in [35] for the linear π-calculus does equate them. Intuitively, this is because the linear type structure allows us to abstract away the additional non-τ-actions in the following way:

1. The action $\overline{y}(e)$ is typed as an output to replication: thus it just replicates a process in the environment without affecting it.
2. The action e is typed as a linear input: hence it necessarily receives its dual output, neither receiving nor emitting non-trivial information.

For these reasons, the additional actions in $[\![(y\star)]\!]_u$ never affect the environment in a way well-typed observers could detect, and are automatically executable, so they behave "as if they were τ-actions", allowing them to be neglected. This suggests the following principle of behavioural semantics in linear processes.

> *Categorise some of the typed actions as "non-affecting", and abstract away non-affecting actions as if they were τ-actions.*

The type structure plays a crucial role in this principle.

Following [7, 11, 14, 18, 33], the linear π-calculus in [35] includes branching/selection, which correspond to sums in the λ-calculus and additives in Linear Logic [12]. A *branching* is an input with I-indexed branches of form $x[\&_{i\in I}(\vec{y}_i).P_i]$, while a *selection* is an output of form $\overline{x}\mathtt{in}_i(\vec{z})Q$. These constructs have the following dynamics: $x[\&_i(\vec{y}_i).P_i]|\overline{x}\mathtt{in}_j(\vec{y}_j)Q \longrightarrow (\boldsymbol{\nu}\,\vec{y}_j)(P_j|Q)$. Now consider another equation in $\lambda_{()\times+}$, which uses sums this time. Let $\mathtt{bool} \stackrel{\text{def}}{=} \mathtt{unit}+\mathtt{unit}$ below.

$$y : \mathtt{bool} \vdash \mathtt{case}\, y\, \mathtt{of}\, \{\mathtt{in}_i() : \mathtt{in}_1(\star)\}_{i\in\{1,2\}} \;=\; \mathtt{in}_1(\star) : \mathtt{bool}$$

These terms are translated as follows:

$$[\![\mathtt{case}\, y\, \mathtt{of}\, \{\mathtt{in}_i() : \mathtt{in}_1(\star)\}_{i\in\{1,2\}}]\!]_u \stackrel{\text{def}}{=} !u(c).\overline{y}(e)e[\&_{1,2}.\overline{c}\mathtt{in}_1]$$
$$[\![\mathtt{in}_1(*)]\!]_u \stackrel{\text{def}}{=} !u(c).\overline{c}\mathtt{in}_1.$$

Both processes are equated by $\cong_\pi$. Intuitively this is because an input at e in the first process surely arrives (due to linearity), and whichever branch is selected it leads to the same selection $\overline{c}\mathtt{in}_1$. We can thus augment the previous principle as follows.

> *We may abstract away linear branching inputs as far as they lead to the same action in all possible branches.*

The precise formulation of this idea is given in Section 2.

Application of Linear Bisimilarity. The new bisimilarity can justify the

equations mentioned above, as well as many of the general equations over linear π-terms in [35] which are used for definability arguments (Proposition 5.7 in [35]). As another application, Section 5 discusses a behavioural theory of secure information flow for the π-calculus, which uses a secrecy-sensitive bisimilarity built on the top of linear bisimilarity. The theory ensures secrecy through semantic means for a strictly larger set of processes than the type-based theory in [23] (which is already powerful enough to embed representative secrecy calculi such as [3, 32]). For example, the theory can justify the safety of the following λ-term by encoding ($\top$ and $\bot$ are high and low secrecy levels, respectively).

$$\mathtt{case}\ y^{\top}\ \mathtt{of}\ \{\mathtt{in}_i() : \mathtt{in}_1(\star)\}_{i\in\{1,2\}} : \mathtt{bool}^{\bot}$$

which is untypable in standard secrecy typing systems, cf. [3, 32].

Summary of Contributions. The following summarises our main technical contributions. To our knowledge the present work is the first to introduce a consistent theory of bisimilarity for the π-calculus which abstracts away non-τ-actions. The secrecy analysis in (3) would also be new.

1. The introduction of a novel bisimilarity for the π-calculus that exploits the linear type structure. While sound, the resulting equivalence is strictly greater than the standard weak bisimilarity.
2. The establishment of congruency of linear bisimilarity in typed contexts. The proof is non-trivial due to the abstraction of non-τ-actions and the use of liveness associated with linearity.
3. An application to secrecy analysis where a secrecy-enhanced version of linear bisimilarity is used for formulating and analysing secure information flow in the π-calculus [20, 23], ensuring secrecy for a strictly greater set of processes than the type-based approaches in [20, 23] while still allowing compositional verification technique.

We also observe that, while the present work concentrates on the pure linear π-calculus, the framework is systematically extendible to more complex type structures which integrate linearity with nontermination, nondeterminism and state [23], to which we can use the same proof methods to obtain the corresponding results. Such extensions are briefly discussed at the end of Section 5.

Related Work. Since the introduction of Linear Logic [12], linearity has been studied in various semantic and syntactic contexts. In the setting of the π-calculus, Kobayashi et al. [25], Yoshida [34] and Sangiorgi [29] studied linearity and its relationship to process equivalences. The present work introduces, for the first time, a consistent theory of bisimilarity based on a labelled transition relation which allows to abstract away non-τ-actions using linear types.

In theories of secure information flow, equality over programs often play a central role, cf. [2, 3, 10, 13, 30]. Among them, Focardi et al. [9, 10] present a bisimulation for cryptographic protocols where high-level actions are abstracted away. In contrast with the present work, [9, 10] are based on CCS without

using type structure. Abadi and his colleagues studied several typing systems and their equivalences for the Spi-calculus and Join calculi in a series of work, e.g. [1, 2, 4]. In particular in [1] Abadi establishes a secrecy theorem based on a may-equivalence by using type information for controlling the interface of the attacker. Abadi, Fournet and Gonthier [5] also study a process calculus with constructs for authentication and show a full abstraction translation of this calculus into a Join-calculus. The main difference with [1, 2, 5] is that we take more abstract approach using linearity of communication in order to limit the environments of opponent processes without specific security constructors such as signatures and nonces, and then apply it for information flow analysis by adding simple security levels to channels. Hennessy and Riely [13] use a secrecy-sensitive may-equivalence for noninterference in the π-calculus. The present work introduces a theory of bisimilarity based on linearity which is directly applicable to secrecy of divergent programming languages. This line of study is not developed in [1, 2, 4, 5, 13]. We also believe our linear bisimilarity technique can be adapted to advanced security constructs such as cryptography [1, 4], authentication [5] and access controls [13] where linearity is often vital. This would allow to verify more protocols compositionally. Finally the first two present authors proposed, in [20] (with Vasconcelos) and in [23], type systems for the π-calculus which ensure secrecy. The present paper gives a semantic theory of secrecy, non-trivially extending the syntactic theories in [20, 23].

Outline of the paper. Section 2 briefly reviews the linear π-calculus in [35]. Section 3 introduces linear bisimilarity, whose congruency proof is given in Section 4. Section 5 discusses an application of the linear bisimilarity to secure information flow analysis. For details of the syntax and types used in the paper, the reader may refer to [7, 35].

Acknowledgements. The authors thank Martin Abadi for his comments on an early version of this paper. The first author is partially supported by by EPSRC grant GR/R33465/01. The second and the third authors are partially supported by EPSRC grant GR/N/37633.

2 Preliminaries

2.1 Processes and Channel Types

The set of *processes* is given by the following grammar [7, 35]. Below and henceforth $x, y, \ldots$ range over a countable set of names.

$$P \ ::= \ x(\vec{y}).P \mid \overline{x}(\vec{y})P \mid x[\&_{i\in I}(\vec{y}_i).P_i] \mid \overline{x}\mathtt{in}_i(\vec{y})P \mid P|Q \ \mid \ (\boldsymbol{\nu}\, x)P \ \mid \ \mathbf{0} \ \mid \ !P.$$

$x(\vec{y}).P$ (resp. $\overline{x}(\vec{y})P$) is a unary input (resp. unary output), while $x[\&_{i\in I}(\vec{y}_i).P_i]$ (resp. $\overline{x}\mathtt{in}_i(\vec{y})P$) is a branching (resp. selection). $P|Q$ is a parallel composition, $(\boldsymbol{\nu}\, x)P$ is a restriction, and $!P$ is a replication. In $!P$ we assume P is either a unary or branching input. The definitions of structural equality $\equiv$, given in Appendix A is standard except for how we ensure output asynchrony. The reduction relation

$\longrightarrow$ is generated from the following rules, closed under output prefix, restriction and parallel composition (modulo $\equiv$).

$$\begin{array}{rl} x(\vec{y}).P|\overline{x}(\vec{y})Q & \longrightarrow (\boldsymbol{\nu}\,\vec{y})(P|Q) \\ !x(\vec{y}).P|\overline{x}(\vec{y})Q & \longrightarrow !x(\vec{y}).P|(\boldsymbol{\nu}\,\vec{y})(P|Q) \\ x[\&_i(\vec{y_i}).P_i]|\overline{x}\mathtt{in}_i(\vec{y_i})Q & \longrightarrow (\boldsymbol{\nu}\,\vec{y_i})(P_i|Q) \\ !x[\&_i(\vec{y_i}).P_i]|\overline{x}\mathtt{in}_i(\vec{y_i})Q & \longrightarrow !x[\&_i(\vec{y_i}).P_i]|(\boldsymbol{\nu}\,\vec{y_i})(P_i|Q) \end{array}$$

Action modes, ranged over by $p, \ldots$, are members of the sets $\{\downarrow, \mathbf{!}\}$ (written $p_{\mathtt{I}}, \ldots$), and $\{\uparrow, \mathbf{?}\}$ (written $p_{\mathtt{O}}, \ldots$). The *dual* $\overline{p}$ of p is given by $\overline{\downarrow} = \uparrow$, $\overline{\mathbf{!}} = \mathbf{?}$ and $\overline{\overline{p}} = p$. Then the set of *channel types* is given by the following grammar. For simplicity we assume indices i range over a fixed set $\{1, 2\}$.

$$\tau \ ::= \ \tau_{\mathtt{I}} \mid \tau_{\mathtt{O}} \mid * \qquad \tau_{\mathtt{I}} \ ::= \ (\vec{\tau_{\mathtt{O}}})^{p_{\mathtt{I}}} \mid [\&_i \vec{\tau_{\mathtt{O}i}}]^{p_{\mathtt{I}}} \qquad \tau_{\mathtt{O}} \ ::= \ (\vec{\tau_{\mathtt{I}}})^{p_{\mathtt{O}}} \mid [\oplus_i \vec{\tau_{\mathtt{I}i}}]^{p_{\mathtt{O}}}$$

Above $\vec{\tau}$ denotes a vector of channel types. We define $\overline{\tau}$, the *dual* of τ, by dualising the action modes and exchanging $\oplus$ and $\&$ of τ. $\mathsf{md}(\tau)$ is $*$ if $\tau = *$, else the outermost action mode of τ. On types $\odot$ is the least commutative partial operation such that (1) $\tau \odot \overline{\tau} = *$ ($\mathsf{md}(\tau) = \downarrow$), (2) $\tau \odot \overline{\tau} = \tau$ ($\mathsf{md}(\tau) = \mathbf{!}$) and (3) $\tau \odot \tau = \tau$ ($\mathsf{md}(\tau) = \mathbf{?}$). A branching type is sometimes written $[\tau_1 \& \tau_2]^p$ and similarly for selection. If $\tau \odot \tau'$ is defined we say they *compose*. Following [7, 22, 35], we assume the following *sequentiality constraint*, which (together with IO-alternation and other elements in the linear type structure) comes from game semantics. We state the constraint only for unary types: for branching/selection types, we require the same constraint for each summand.

- In $(\vec{\tau})^{\downarrow}$, $\mathsf{md}(\tau_i) = \mathbf{?}$ for each $1 \leq i \leq n$. Dually for $(\vec{\tau})^{\uparrow}$.
- In $(\vec{\tau})^{\mathbf{!}}$ $\mathsf{md}(\tau_i) \in \mathbf{?}$ for each $1 \leq i \leq n$ except at most one j for which $\mathsf{md}(\tau_j) = \uparrow$. Dually for $(\vec{\tau})^{\mathbf{?}}$.

2.2 Typing and Typed Processes

An *action type* is a finite acyclic directed graph whose nodes have the form $x : \tau$ such that no names occur twice and each edge is of form $x : \tau \to x' : \tau'$ with either $\mathsf{md}(\tau) = \downarrow$ and $\mathsf{md}(\tau') = \uparrow$, or $\mathsf{md}(\tau) = \mathbf{!}$ and $\mathsf{md}(\tau') = \mathbf{?}$. We write $A(x)$ for the channel type assigned to x occurring in A. The partial operator $A \odot B$ is defined iff channel types in common names compose and the adjoined graph do not have a cycle. If so, the result is a graph in which intermediate edges are taken away from the adjoined graph (see Appendix B for detailed formal definitions). To avoid divergence, this operator ensures that processes never exhibit circular dependency in actions. For example, $x : \tau_1 \to y : \tau_2$ and $y : \overline{\tau_2} \to x : \overline{\tau_1}$ are not composable. $\mathsf{fn}(A)$ and $\mathsf{md}(A)$ denote the sets of free names and modes in A, respectively. $A \asymp B$ indicates $A \odot B$ is defined.

Sequents of the linear typing have the form $\vdash P \triangleright A$.[1] The rules are given in Appendix C. The system is identical with [35] except that we assume a linear

[1] In [35] we used a different main sequent, $\Gamma \vdash P \triangleright A$. This is equivalent to the present one by adjoining Γ to the right-hand side.

input does not suppress more than one linear output.[2] If $\vdash P \triangleright A$ is derivable, we say P *is typable with* A. For brevity we sometimes write P^A instead of $\vdash P \triangleright A$. Typable processes are often called *linear processes*.

Example 1. (linear processes)

1. $\vdash \overline{x} \triangleright x : ()^{\uparrow}$ and $\vdash x.\mathbf{0}|\overline{x} \triangleright x : *$. The former can also be typed with $x : ()^{?}$.
2. $\vdash !u(c).\overline{c} \triangleright u : (()^{\uparrow})^{!}$ and $\vdash !u(c).\overline{x}(e)e.\overline{c} \triangleright u : (()^{\uparrow})^{!} \to x : (()^{\downarrow})^{?}$.
3. Let $\mathbb{B} = [\varepsilon \oplus \varepsilon]^{\uparrow}$ (where ε is the empty vector). Then $\vdash !u(c).\overline{x}(e)e[.\overline{c}\mathtt{in}_1 \& .\overline{c}\mathtt{in}_2] \triangleright u : (\mathbb{B})^{!} \to x : (\overline{\mathbb{B}})^{?}$. Other terms typable with this type include $!u(c).\overline{c}\mathtt{in}_1$ and $!u(c).\overline{x}(e)e[.\overline{c}\mathtt{in}_1 \& .\overline{c}\mathtt{in}_1]$ as well as their symmetric variants.

The following properties of typed terms are from [35]. (3) is a consequence of strong normalisability of linear processes and will play an important role later.

Proposition 1.

1. (subject reduction) *If* $\vdash P \triangleright A$ *and* $P \longrightarrow Q$ *then* $\vdash Q \triangleright A$.
2. (one-step confluence) *If* $\vdash P \triangleright A$ *and* $P \longrightarrow Q_i$ $(i = 1, 2)$ *then* $Q_1 \equiv Q_2$ *or* $Q_i \longrightarrow R$ $(i = 1, 2)$ *for some* R.
3. (liveness) *Let* $\vdash P \triangleright A \otimes x : \tau$ *with* $\mathsf{md}(\tau) = \uparrow$ *and* $\mathsf{md}(A) \subseteq \{!, *\}$ *($\otimes$ is the graph union). Then* $P \longrightarrow^{*} P'$ *such that* $P' \equiv \overline{x}(\vec{y})R$ *or* $P' \equiv \overline{x}\mathtt{in}_j(\vec{y})R$.

2.3 Contextual Congruence and Bisimilarity

A relation $\mathcal{R}$ over typed processes is *typed* when $P_1^{A_1} \,\mathcal{R}\, P_2^{A_2}$ implies $A_1 = A_2$. We write $P_1 \,\mathcal{R}^A\, P_2$ when P_1^A and P_2^A are related by a typed relation $\mathcal{R}$. A *typed congruence* is a typed relation which is an equivalence closed under all typed contexts. The *contextual congruence* $\cong_\pi$ is the maximum typed congruence satisfying the following condition ($\mathbb{B}$ appeared in Example 1).

$$\text{If } P \Downarrow_x^i \text{ and } P \cong_\pi^{x:\mathbb{B}} Q, \text{ then } Q \Downarrow_x^i \qquad (i = 1, 2)$$

where $P \Downarrow_x^i$ means $P \longrightarrow^{*} \overline{x}\mathtt{in}_i(\vec{y})P'$. The relation is maximally consistent in the sense that any addition of equations leads to inconsistency.

The congruence $\cong_\pi$ may be considered as giving the maximal meaningful way to equate processes. A more restricted and more tractable equality is obtained using labelled transition. Let $l, l', \ldots$ be given by:

$$l \;::=\; \boldsymbol{\tau} \mid x(\vec{y}) \mid \overline{x}(\vec{y}) \mid x\mathtt{in}_i(\vec{y}) \mid \overline{x}\mathtt{in}_i(\vec{y})$$

$\mathsf{bn}(l)$ denotes bound names in l. If $l \neq \boldsymbol{\tau}$, we write $\mathsf{sbj}(l)$ for the initial free name of l. Using these labels, the typed transition $P^A \xrightarrow{l} Q^B$ is defined as in Appendix D. The weak bisimilarity induced by the transition is denoted $\approx$.

As indicated in the introduction, $\cong_\pi$ is strictly greater than $\approx$. One of the aims of the present work is to fill the gap between $\approx$ and $\cong_\pi$, at least partially, without loosing the ease of reasoning of $\approx$.

[2] This condition, which we call *unique-answer-per-thread*, is essentially the same condition as the one used for sequentialisation in Proposition 5.6 in [35], and leads to a more elegant technical development of the theory of linear bisimilarity without sacrificing basic expressiveness of the calculus.

3 Linear Bisimilarity

3.1 Categorising Actions

We begin our path towards the definition of linear bisimilarity with classifying types according to the following criteria: whether typable actions affect the environment non-trivially; and whether these actions are guaranteed to take place.

Definition 1. (affecting and enabled types)

1. τ is *affecting* iff there exist $\vdash P_{1,2} \triangleright x : \tau$ and a typed context $C[\,\cdot\,]$ such that $\vdash C[P_i] \triangleright u : \mathbb{B}$, $C[P_1] \Downarrow_u^1$ and $C[P_2] \Downarrow_u^2$.
2. τ is *enabling* iff $\vdash P \triangleright x : \tau$ implies $P \longrightarrow^* P' \xrightarrow{l}$ such that $\mathsf{sbj}(l) = x$. τ is *enabled* if $\overline{\tau}$ is enabling.

Example 2. (affecting and enabled types)

1. $\mathbb{B}$, $(\mathbb{B})^{\mathbf{!}}$ and $((\mathbb{B})^{\mathbf{!}})^{\uparrow}$ are affecting but $((\mathbb{B})^{\mathbf{!}})^{\mathbf{?}}$ and $(((\mathbb{B})^{\mathbf{!}})^{\mathbf{?}})^{\downarrow}$ are not. It is notable that no τ such that $\mathsf{md}(\tau) \in \{\mathbf{?}, \downarrow\}$ is affecting.
2. Any τ such that $\mathsf{md}(\tau) \in \{\downarrow, \uparrow, \mathbf{!}\}$ is enabling, while any τ such that $\mathsf{md}(\tau) = \mathbf{?}$ is not. Hence all and only types that are enabled are those with outermost modes $\downarrow, \uparrow, \mathbf{?}$.

As suggested in the above example, we have an easy rule to determine whether a type is affecting or not, based on the shape of types.

Proposition 2. *Define* Aff *as the smallest set of types satisfying the following conditions.*

- $[\oplus_{i=1,2}\vec{\tau_i}]^{\uparrow} \in \mathsf{Aff}$.
- $(\tau_1..\tau_n)^{\uparrow} \in \mathsf{Aff}$ *when* $\tau_i \in \mathsf{Aff}$ *for some* i $(1 \leq i \leq n)$.
- $(\tau_1..\tau_n)^{\mathbf{!}}$ *when* $\tau_i \in \mathsf{Aff}$ *for some* i $(1 \leq i \leq n)$.
- $[\&_{i=1,2}\tau_{i1}..\tau_{in_i}]^{\mathbf{!}} \in \mathsf{Aff}$ *when* $\tau_{ij} \in \mathsf{Aff}$ *for some* i *and* j $(i \in \{1,2\},\ 1 \leq j \leq n_i)$.

Then τ *is affecting iff* $\tau \in \mathsf{Aff}$.

Note that τ such that $\mathsf{md}(\tau) \in \{\mathbf{?}, \uparrow\}$ is never in Aff by definition. Below and henceforth we say P is *prime with subject* x if either (1) P is input-prefixed with subject x or (2) P has form $\overline{x}(y_1..y_n)\Pi_i R_i$ or $\overline{x}\mathtt{in}_i(y_1..y_n)\Pi_i R_i$ where each R_i is prime with subject x.

Proof. **($\tau \in \mathsf{Aff}$ implies τ affecting)** By induction on the generation rules above. We use the following fact: to differentiate $\vdash P_{1,2} \triangleright x : \tau$, we can always choose a context $(\boldsymbol{\nu}\, x)(R|[\,])$ where R is a prime term with subject x, typed as $\overline{\tau}$, such that $(\boldsymbol{\nu}\, x)(R|P_1)$ is distinguishable from $(\boldsymbol{\nu}\, x)(R|P_2)$. This is a consequence of the Context Lemma and, in addition, operational reasoning based on the type structure, as noted in [35] for linear processes. A detailed proof for affine types is given in [7]. The case of $[\oplus_{i=1,2}\vec{\tau_i}]^{\uparrow} \in \mathsf{Aff}$ is obvious. Of the remaining cases we only present one as the others are similar. Take, for simplicity, $(\tau)^{\mathbf{!}}$ with τ

affecting. Take the context $(\boldsymbol{\nu}\, x)(R|[\,])$ which differentiates between $\vdash P_{1,2} \triangleright x : \tau$. Now let $P'_i \stackrel{\text{def}}{=} !u(x).P_i$ $(i = 1, 2)$. Clearly $\vdash P'_{1,2} \triangleright x : (\tau)^{\mathbf{!}}$. These agents can now be distinguished by $(\boldsymbol{\nu}\, u)(\overline{u}(x)R|[\,])$, showing $(\tau)^{\mathbf{!}}$ is affecting.

(τ affecting implies $\tau \in \mathsf{Aff}$) We establish the contrapositive by noting that the complement of Aff is generated in a similar way. ■

Remark 1. It is worth noting that, in the following technical development, we may as well classify all linear unary types as affecting, with no change in the resulting bisimilarity. However the behavioural notion of affecting types in Definition 1 motivates our construction on a uniform basis.

The classification of types given above induces the classification of actions. First, an action annotated with an action type, say l^A, is called a *typed action* if the shape of l conforms to A. For example, if $l = \overline{x}\mathtt{in}_1$ then l^A is a typed action iff $A(x) = \mathbb{B}$. $\boldsymbol{\tau}^A$ is a typed action for an arbitrary A. If $l \neq \boldsymbol{\tau}$ and l^A is typed, the *type of* l^A is $A(\mathsf{sbj}(l))$. Then we say:

Definition 2. (affecting and enabled typed actions) l^A is *affecting* if $l \neq \boldsymbol{\tau}$ and the type of l^A is affecting; l^A is *non-affecting* if it is not affecting. Further l^A is *enabled* if $l = \boldsymbol{\tau}$ or the type of l^A is enabled.

Table 1 illustrates how typed actions are classified, writing $\boldsymbol{\tau}$, $\downarrow()$, $\uparrow()$, $\downarrow\&$, $\uparrow\oplus$, **!** and **?** for (respectively) the $\boldsymbol{\tau}$-action, unary linear input, unary linear output, linear branching, linear selection, replicated unary/branching input, and its dual output.[3]

	$\boldsymbol{\tau}$	$\downarrow()$	$\uparrow()$	$\downarrow\&$	$\uparrow\oplus$	**!**	**?**
affecting	no	no	yes	no	yes	yes	no
enabled	yes	yes	yes	yes	yes	no	yes

Table 1. Classification of Actions

We can now introduce invisibility under linear type structure which dictates the "$\boldsymbol{\tau}$-like" nature of certain non-$\boldsymbol{\tau}$-actions in the typed setting. Below and henceforth $\Delta, \Gamma, \ldots$ range over finite sets of names. $\mathsf{fn}(l)$ is the set of free names in l while $\mathsf{bn}(l)$ is the set of free names in l.

Definition 3.

1. (invisible actions) A typed action l^A is Δ-*invisible* (Δ-i.) when either $\mathsf{fn}(l) \cap \Delta = \emptyset$ or, if not, l^A is an output which is non-affecting.[4] If l^A is Δ-invisible and, moreover, is enabled, then l^A is Δ-*strongly invisible* (Δ-s.i.).

[3] We classify unary linear outputs $\uparrow()$ as affecting in Table 1 even though they are sometimes not, as is seen from Proposition 2. An example is $()^{\uparrow}$.

[4] We can consistently abstract away linear input at Δ. For simplicity and because this may not significantly change the resulting relation, we use the present definition.

2. (abstracted transitions) $P^A \xrightarrow{\hat{l}}_\Delta Q^B$ when either: (1) $P^A \xrightarrow{l} Q^B$ or (2) $B = A$, $Q = P$ and l^A is Δ-invisible. $P^A \Longrightarrow_\Delta Q^B$ denotes $P^A \xrightarrow{l_1 \ldots l_n} Q^B$ ($n \geq 0$) where each l_i is strongly Δ-invisible; then $P^A \overset{l}{\Longrightarrow}_\Delta Q^B$ denotes $P^A \Longrightarrow_\Delta \xrightarrow{l}_\Delta \Longrightarrow_\Delta Q^B$; finally $P^A \overset{\hat{l}}{\Longrightarrow}_\Delta Q^B$ denotes either $P^A \overset{l}{\Longrightarrow}_\Delta Q^B$ or $P^A \Longrightarrow_\Delta Q^B$ where l is invisible and $\mathsf{fn}(B) \cap \mathsf{bn}(l) = \emptyset$. If $P^A \Longrightarrow_\Delta Q^B$ is induced by $P^A \xrightarrow{l_1 \ldots l_n} Q^B$, we say the latter *underlies* the former.

Note that the standard abstracted transitions are a special case of those defined above. For example, $P^A \overset{\hat{l}}{\Longrightarrow}_\Delta Q^B$ means $P^A \xrightarrow{l_1..l_n} \xrightarrow{\hat{l}}_\Delta \xrightarrow{l'_1..l'_m} Q^B$ for some Δ-strong invisible $l_1..l_n$ and $l'_1..l'_m$; if we restrict all of $l_{1..n}$ and $l'_{1..m}$ to $\boldsymbol{\tau}$-actions, and use $\xrightarrow{\hat{l}}$ instead of $\xrightarrow{\hat{l}}_\Delta$, then we obtain the standard notion of abstracted transition. Note also there may be more than one sequences of non-$\boldsymbol{\tau}$-actions which underly a given abstracted transition.

3.2 Semi-typed Relation and Branching Closure

The invisibility of non-τ-actions necessitates one fundamental change in the notion of bisimulation. As an illustration we go back to the initial example in the introduction. The two typed processes concerned were $!x(c).\overline{c}^A$ and $!x(c).\overline{y}(e).\overline{c}^A$ with $A = x : ((\,)^\uparrow)^! \to y : ((\,)^\downarrow)^?$. After the common initial action, the typing becomes $A \otimes c : (\,)^\uparrow$. But if $\overline{y}(e).\overline{c}^A$ has an output action (which should and can be abstracted away), then e becomes free in the residual and appears in its type environment. This state should be related to the other process which still has type $A \otimes c : (\,)^\uparrow$. Consequently, a bisimulation needs to relate processes with distinct action types.

Definition 4. A relation $\mathcal{R}$ on typed processes is *semi-typed* when $P^A \mathcal{R} Q^B$ implies that the projections of A and B on $\mathsf{fn}(A) \cap \mathsf{fn}(B)$ coincide. We write $P^A \mathcal{R}^\Delta Q^B$ if $\mathcal{R}$ is semi-typed and $\mathsf{fn}(A) \cap \mathsf{fn}(B) = \Delta$, in which case we say P^A *and* Q^B *are related by* $\mathcal{R}$ *at* Δ. The maximum typed subrelation of a semi-typed $\mathcal{R}$ is called its *centre*.

Using semi-typed relation, a natural way to define a bisimulation would be as follows: a semi-typed $\mathcal{R}$ such that, whenever $P_1^{A_1} \, \mathcal{R} \, P_2^{A_2}$ with $\Delta = \mathsf{fn}(A_1) \cap \mathsf{fn}(A_2)$, we have the following and its symmetric case:

$$\text{whenever } P_1^{A_1} \xrightarrow{l} Q_1^{B_1}, \text{ there is } P_2^{A_2} \overset{\hat{l}}{\Longrightarrow}_\Delta Q_2^{B_2} \text{ such that } Q_1^{B_1} \, \mathcal{R} \, Q_2^{B_2}.$$

However the following example shows that congruency is lost if we allow branching.

Example 3. $\overline{x}\mathtt{in}_1^{x:\mathbb{B}}$ and $\overline{y}\mathtt{in}_1^{y:\mathbb{B}}$ are bisimilar at $\emptyset$ in the above definition. Similarly $x[\&_{1,2}\overline{z}\mathtt{in}_i]$ and $y[\&_{1,2}\overline{z}\mathtt{in}_i]$ are bisimilar at z. However when we compose them in pairs, $(\overline{x}\mathtt{in}_1 | x[\&_{1,2}\overline{z}\mathtt{in}_i])$ and $(\overline{y}\mathtt{in}_2 | y[\&_{1,2}\overline{z}\mathtt{in}_i])$ are *not* bisimilar: in fact these terms can be regarded as, up to redundant reduction, $\overline{z}\mathtt{in}_1$ and $\overline{z}\mathtt{in}_2$, which would not be equated under any reasonable semantic criteria.

The problem in this example is in the second equation: intuitively, $x[\&_{1,2}\overline{z}\mathtt{in}_i]$ and $y[\&_{1,2}\overline{z}\mathtt{in}_i]$ cannot be equated because, at the disparate interfaces (here x and y), we should expect anything can happen: thus it is possible, at x, the first process receives the left selection, while, at y, the second process receives the right selection (which is precisely what happens in the composition). This indicates that we should say "for all possible branching at disparate channels, the behaviours of two processes at common channels coincide." This idea is formalised in the following definition. Below $t, t_i, \ldots$ range over sequences of typed transitions. A *branching variant* of, say, $x\mathtt{in}_1(\vec{y})$ is $x\mathtt{in}_2(\vec{z})$ (conforming to the given typing), taken up to α-equality.

Definition 5. (branching closure) A set $\{P^A \xrightarrow{t_i} Q_i^{B_i}\}_{i\in I}$ of sequences of typed transitions is Δ-*branching closed* (Δ-b.c.) iff: whenever $t_i = sls' \in S$ with l being a linear branching input such that $\mathsf{fn}(l) \cap \mathsf{fn}(\Delta) = \emptyset$, there is $t_j = sl's''$ $(j \in I)$ for each branching variant l' of l.

Accordingly we say $\{P^A \stackrel{\hat{l}}{\Longrightarrow}_\Delta Q_i^{B_i}\}_{i\in I}$ is Δ-branching closed if there exists a Δ-branching closed set $\{P^A \xrightarrow{t_i} Q_i^{B_i}\}_{i\in I}$ where $P^A \xrightarrow{t_i} Q_i^{B_i}$ underlies $P^A \stackrel{\hat{l}}{\Longrightarrow}_\Delta Q_i^{B_i}$ for each i. Similarly for other forms of abstracted transitions.

3.3 Linear Bisimulation

We can now introduce a new bisimilarity on linear processes, which we call linear bisimilarity.

Definition 6. (linear bisimulation) A semi-typed $\mathcal{R}$ is a *linear bisimulation* when $P_1^{A_1}\,\mathcal{R}\,P_2^{A_2}$ with $\Delta = \mathsf{fn}(A_1) \cap \mathsf{fn}(A_2)$ implies the following and its symmetric case: whenever $P_1^{A_1} \xrightarrow{l} Q_1^{B_1}$, there is a Δ-closed $\{P_2^{A_2} \stackrel{\hat{l}}{\Longrightarrow}_\Delta Q_{2i}^{B_{2i}}\}_{i\in I}$ such that $Q_1^{B_1}\,\mathcal{R}\,Q_{2i}^{B_{2i}}$ for all $i \in I$. The maximum linear bisimulation exists, denoted $\approx_{\mathrm{L}}$.

Simple examples of (non-)bisimilarity follow. Below and henceforth we omit obvious type annotations, assuming all processes are well-typed. We often annotate $\approx_{\mathrm{L}}$ as $\approx_{\mathrm{L}}^{x,y}$ (which follows Definition 4) to make intersecting channels explicit.

Example 4. 1. $x.\mathbf{0} \approx_{\mathrm{L}}^{\emptyset} \mathbf{0}$ and $!x.\mathbf{0} \approx_{\mathrm{L}}^{\emptyset} \mathbf{0}$ and $\overline{x}|\overline{x} \approx_{\mathrm{L}}^{x} \overline{x} \approx_{\mathrm{L}}^{x} \mathbf{0}$.
2. $x.\overline{y}\mathtt{in}_1 \approx_{\mathrm{L}}^{y} \overline{y}\mathtt{in}_1$. Intuitively this is because an output at x will surely arrive in which case the former process has the same observable as the latter.
3. Because of the lack of branching closure, we have $x[\&_{1,2}\overline{z}\mathtt{in}_i] \not\approx_{\mathrm{L}}^{z} y[\&_{1,2}\overline{z}\mathtt{in}_i]$. On the other hand, we have $x[\&_{1,2}\overline{z}\mathtt{in}_1] \approx_{\mathrm{L}}^{z} y[\&_{1,2}\overline{z}\mathtt{in}_1] \approx_{\mathrm{L}}^{z} \overline{z}\mathtt{in}_1$.

We prove the following result in the next section.

Theorem 1. *The centre of $\approx_{\mathrm{L}}$ is a congruence.*

Since an action of $\mathbb{B}$-type is always visible, we immediately obtain:

Corollary 1. *The centre of $\approx_{\mathrm{L}}$ is a subrelation of $\cong_{\pi}$.*

We give simple applications of the linear bisimilarity. Below (1) says processes which are entirely typed with **?**-types are equated with the inaction (pA means $\mathsf{md}(A) = \{p\}$). (2) says there is essentially a unique inhabitant in the unit type of λ-calculi. (3) uses (2) to derive the equality over `unit`-type $\lambda_{()\times+}$-terms.

Proposition 3.

1. (innocuous actions, cf. [20]) *If $\vdash P \triangleright \mathbf{?}A$ then $P \approx_{\mathrm{L}}^{\mathsf{fn}(A)} \mathbf{0}$.*
2. (unit inhabitation, 1) *$\vdash P \triangleright A$ with $A = x : ((\,)^{\uparrow})^{!} \to A_0$ implies $P \approx_{\mathrm{L}} {}^{x}!x(c).\overline{c}$.*
3. (unit inhabitation, 2) *If $\Gamma \vdash M : \mathtt{unit}$ in $\lambda_{()\times+}$ then $\Gamma \vdash M \cong \star : \mathtt{unit}$ where $\cong$ is the standard contextual equivalence in $\lambda_{()\times+}$.*

Proof. For (1), let $\mathcal{R}$ be a semi-typed relation given by: $P^{A}\mathcal{R}^{\Gamma}\mathbf{0}$ iff (i) $\Gamma \subset \mathsf{fn}(A)$ and (ii) $x \in \Gamma$ implies $\mathsf{md}(A(x)) = \mathbf{?}$. Assume $P^{A} \xrightarrow{l} Q^{B}$. If $l \neq \boldsymbol{\tau}$ and $\mathsf{sbj}(l) \in \Gamma$ then the type of l^{A} is **?**-mode, so l^{A} is Γ-invisible. If not, obviously l^{A} is Γ-invisible. So in both cases $\mathbf{0}$ simulates this by the inaction. Since Γ-part of B is identical with that of A, we are done. For (2), by $P \equiv !x(c).P' | R$ with $\vdash R \triangleright \mathbf{?}A_0$, (1) and Theorem 1, we can safely ignore R. So assume $P \equiv !x(c).P'$. Again by Theorem 1 it is enough to show $P' \approx_{\mathrm{L}}^{c} \overline{c}$. Since all actions on both sides are invisible we are done. (3) is immediate from (2) and Theorem 1 together with Theorem 5.9 (full abstraction) in [35]. ■

Other applications include a simple proof of the equations for sequentialisation used in [35], as well as a behavioural theory of secrecy which we develop in Section 5.

4 Properties of Linear Bisimilarity

4.1 Preparation

The purpose of this section is to establish congruency of (the centre of) $\approx_{\mathrm{L}}$. We shall use the following basic properties of typed transitions. The proofs are straightforward and are omitted. Below in (1, 2), the *index* of an action is the least subterm(s) that the action originates from (cf. [7, Appendix F]).

Proposition 4.

1. *If $P^{A} \xrightarrow{l_i} Q_i^{B_i}$ $(i = 1, 2)$ with $l_1 = \boldsymbol{\tau}$ and $l_2 \neq \boldsymbol{\tau}$. Then the indices of these two actions are distinct.*
2. (confluence) *Let $P^{A} \xrightarrow{l_i} Q_i^{B_i}$ $(i = 1, 2)$ such that the indices of these two actions are distinct. Then $Q_1^{B_1} \xrightarrow{l_2} R^{C}$ and $Q_2^{B_2} \xrightarrow{l_1} R^{C}$ for some R^{C}.*
3. (input availability) *Let $\mathsf{md}(A(x)) \in \{\downarrow, \mathbf{!}\}$. Then $P^{A} \xrightarrow{l}$ where l is input with subject x. Hence if $P^{A} \stackrel{l}{\Longrightarrow}_{\Delta} Q^{B}$ with l input then $P^{A} \xrightarrow{l} \Longrightarrow_{\Delta} Q^{B}$.*
4. (strengthening) *If l^{A} is Γ-(s.)i. and $\Delta \subseteq \Gamma$ then l^{A} is Δ-(s.)i. Hence if $P^{A} \stackrel{\hat{l}}{\Longrightarrow}_{\Gamma} Q^{B}$ and $\Delta \subseteq \Gamma$ with $\mathsf{fn}(l) \subset \mathsf{fn}(\Delta)$, then $P^{A} \stackrel{\hat{l}}{\Longrightarrow}_{\Delta} Q^{B}$.*
5. (weakening) *Let $P^{A} \xrightarrow{l}$ and $x \notin \mathsf{fn}(P) \cup \Gamma$. Then if l^{A} is Γ-(s.)i, then l^{A} is $\Gamma \cdot x$-(s.)i. Hence if $P^{A} \stackrel{\hat{l}}{\Longrightarrow}_{\Gamma} Q^{B}$ and $x \notin \mathsf{fn}(P) \cup \Gamma$, then $P^{A} \stackrel{\hat{l}}{\Longrightarrow}_{\Gamma \cdot x} Q^{B}$.*
6. (disjointness) *$P^{A} \approx_{\mathrm{L}} Q^{B}$ for any P^{A}, Q^{B} such that $\mathsf{fn}(A) \cap \mathsf{fn}(B) = \emptyset$.*

4.2 Bisimulation over Unary Processes

We first summarise a property of linear bisimulation over unary processes, i.e. processes without branching or selection. We denote this subset of linear processes by $\mathbb{P}_{()}$ and use $\approx_{\mathrm{L}()}$ for the linear bisimilarity over $\mathbb{P}_{()}$. In the following, $A_1 \bowtie A_2$ means $A_1 \restriction \Gamma = A_2 \restriction \Gamma$ where $\Gamma = \mathsf{fn}(A_1) \cap \mathsf{fn}(A_2)$.

Proposition 5. *Let* $\vdash P_i \triangleright A_i$ $(i = 1, 2)$ *such that* $A_1 \bowtie A_2$. *Then* $P_1^{A_1} \approx_{\mathrm{L}()} P_2^{A_2}$.

Proof. We show the set of such pairs is a bisimulation. First we note that $P^A \xrightarrow{l} Q^B$ implies $A \bowtie B$. Take $P_1^{A_1}$ and $P_2^{A_2}$ as above and let $\Delta = \mathsf{fn}(A_1) \cap \mathsf{fn}(A_2)$. Let $P_1^{A_1} \xrightarrow{l} Q_1^{B_1}$. If l is invisible this is emulated by inaction. If l is visible, then (by P_1 being unary) l is an replicated input at Δ, with $B_1 = A_1 \otimes C$ for some C. By $A_1 \bowtie A_2$ and Proposition 4 (3), we know $P_2^{A_2} \xrightarrow{l} Q_1^{A_2 \otimes C}$. Since $A_1 \otimes C \bowtie A_2 \otimes C$, we are done.

Using this lemma, it is easy to prove the compatibility of $\approx_{\mathrm{L}()}$ under all typed operators since $\bowtie$-related processes always result in $\bowtie$-related processes by each typed operation (for example, if $\vdash P_i \triangleright A_i$ and $\vdash Q_i \triangleright B_i$ with $A_i \asymp B_i$, $A_1 \bowtie A_1$ and $B_1 \bowtie B_1$, then we have $\vdash P_1 \mid Q_1 \triangleright A_1 \odot B_1$, $\vdash P_2 \mid Q_2 \triangleright A_2 \odot B_2$ and $(A_1 \odot B_1) \bowtie (A_2 \odot B_2)$). Hence we have:

Corollary 2. *1. Let* $\vdash P_i \triangleright A$ $(i = 1, 2)$. *Then* $P_1^A \approx_{\mathrm{L}()} P_2^A$.
2. $\approx_{\mathrm{L}()}$ *is a congruence over* $\mathbb{P}_{()}$, *and* $\approx_{\mathrm{L}()} = \cong_\pi$ *in* $\mathbb{P}_{()}$.

This corollary says that *any two unary processes of the same type are semantically equal*. This is reminiscent of the following property of a simply typed λ-calculus: any two terms of a type generated from the unit type by applying the function space and the product zero or more times, are semantically equal (which in turn corresponds to the fact that all objects generated from a final object by product and exponentiation are again final).

4.3 Congruency of $\approx_{\mathrm{L}}$ (1)

As seen in Example 4, Proposition 5 is not valid in the full calculus (e.g. $\overline{x}\mathtt{in}_1 \not\approx_{\mathrm{L}} \overline{x}\mathtt{in}_2$). The proof for a closure of the full calculus is non-trivial due to the strong invisibility and branching-closure. First, we state the following lemma related to the branching-closedness.

Lemma 1.

1. (concatenation) *Assume* $\{P \stackrel{\hat{l_1}}{\Longrightarrow}_\Gamma P'_k\}$ *and* $\{P'_k \stackrel{\hat{l_2}}{\Longrightarrow}_\Gamma P''_{j_k}\}$ *are* Γ*-b.c. Then* $\{P \stackrel{\hat{l_1} \cdot \hat{l_2}}{\Longrightarrow}_\Gamma P'_{j_k}\}$ *are* Γ*-b.c.*
2. (split) *Assume* $\{P \Longrightarrow_\Gamma P_k \xrightarrow{l} P'_k \Longrightarrow_\Gamma P''_{k_j}\}$ *is* Γ*-b.c. Then* $\{P \Longrightarrow_\Gamma P_k\}$ *and* $\{P'_k \Longrightarrow_\Gamma P''_{k_j}\}$ *are* Γ*-b.c.*

3. (restriction) $\{P^A \stackrel{\hat{l}}{\Longrightarrow}_{\Delta\cdot x} Q_i^{B_i}\}$ *is* $\Delta \cdot x$*-b.c. with* $\mathsf{sbj}(l) \neq x$ *and* $\mathsf{md}(A(x)) \in \{\mathbf{!}, *\}$ *iff* $\{(\boldsymbol{\nu}\, x)P^{A/x} \stackrel{\hat{l}}{\Longrightarrow}_{\Delta} (\boldsymbol{\nu}\, x)Q_i^{B_i/x}\}$ *is* Δ*-b.c.*

Proof. **(1)** Obvious by definition.

(2) Suppose $\{P \Longrightarrow_\Gamma P_k \stackrel{l}{\longrightarrow} P'_k \Longrightarrow_\Gamma P''_{k_j}\}$ is Γ-b.c. Assume towards a contradiction that $\{P \Longrightarrow_\Gamma P_k\}_{k\in K}$ is not a Γ branching-closed. Suppose $P \stackrel{l_{k,1}\cdots l_{k,n}}{\longrightarrow} P_k$ underlines $P \Longrightarrow_\Gamma P_k$. Then there is a transition sequence $P \stackrel{l_{k+1,1}\cdots l_{k+1,n}}{\longrightarrow} P_{k+1}$ such that $l_{k+1,m}$ is a branching variant of $l_{j,m}$ for some $1 \leq m \leq n$ and $j \in K$. But this contradicts the assumption that $\{P \Longrightarrow_\Gamma P_k \stackrel{l}{\longrightarrow} P'_k \Longrightarrow_\Gamma P''_{k_j}\}$ is branching closed. We can similarly prove branching-closure of $\{P'_k \Longrightarrow_\Gamma P''_{k_j}\}$ for each k.

(3) Mechanical by the definition of $\stackrel{l}{\longrightarrow}$.

Now we show that $\approx_{\mathrm{L}}$ is transitive on its centre.

Proposition 6. (transitivity) *Suppose* $P_1^{A_1} \approx_{\mathrm{L}}^{\Gamma} P_2^{A_2}$ *and* $P_2^{A_2} \approx_{\mathrm{L}}^{\Delta} P_3^{A_3}$ *such that* $\mathsf{fn}(A_1) \cap \mathsf{fn}(A_3) = \Gamma \cap \Delta$*. Then* $P_1^{A_1} \approx_{\mathrm{L}}^{\Gamma\cap\Delta} P_3^{A_3}$*.*

Proof. The proof uses Proposition 4 (4) and Lemma 1 (1). We show the generated relation to be a bisimulation. Let $P_1^{A_1} \approx_{\mathrm{L}}^{\Gamma} P_2^{A_2}$ and $P_2^{A_2} \approx_{\mathrm{L}}^{\Delta} P_3^{A_3}$ s.t. $\mathsf{fn}(A_1) \cap \mathsf{fn}(A_3) = \Gamma \cap \Delta$. Let $P_1^{A_1} \stackrel{l}{\longrightarrow} Q_1^{B_1}$. By assumption and by Lemma 1 (1):

$$\{P_2^{A_2} \stackrel{l_1\cdots l_n}{\longrightarrow} \stackrel{\hat{l}}{\longrightarrow}_{\Delta} \stackrel{l'_1\cdots l'_m}{\longrightarrow} Q_{2i}^{B_{2i}}\} \quad \text{with} \quad Q_{2i}^{B_{2i}} \approx_{\mathrm{L}} Q_1^{B_1}$$

where each of l_i and l'_j is strongly Δ-invisible. Again by assumption:

$$\{P_3^{A_3} \stackrel{\hat{l_1}}{\Longrightarrow}_\Gamma .. \stackrel{\hat{l_n}}{\Longrightarrow}_\Gamma \stackrel{\hat{l}}{\longrightarrow}_\Gamma \stackrel{\hat{l'_1}}{\Longrightarrow}_\Gamma .. \stackrel{\hat{l'_m}}{\Longrightarrow}_\Gamma Q_{3i_k}^{B_{3i_k}}\} \quad \text{with} \quad Q_{3i_k}^{B_{3i_k}} \approx_{\mathrm{L}} Q_{2i}^{B_{2i}}$$

Note that this is again Δ-b.c. by Lemma 1 (1). We now observe that, by Proposition 4 (4), strongly Γ-invisible actions involved in each of $\stackrel{\hat{l_i}}{\Longrightarrow}_\Gamma$ and $\stackrel{\hat{l'_j}}{\Longrightarrow}_\Gamma$ are all strongly $\Gamma \cap \Delta$-invisible. Similarly l_i and l'_j are (strongly Δ-invisible hence) strongly $\Gamma \cap \Delta$-invisible. Thus we know

$$\{P_3^{A_3} \stackrel{\hat{l}}{\Longrightarrow}_{\Gamma\cap\Delta} Q_{3i_k}^{B_{3i_k}}\} \quad \text{with} \quad Q_{3i_k}^{B_{3i_k}} \approx_{\mathrm{L}} Q_{2i}^{B_{2i}}$$

as required. ■

Remark 2. Proposition 6 does not hold in general if the side condition on free names in the statement (which comes from the renaming theory over processes in [16]) is taken off. In the light of theory in [16], we may as well require semi-typed relations to be renaming closed, in which case this side condition is not restrictive when, for example, we compose two semi-typed relations. The theories [15, 16] also gives a basic framework for treating composability of two typed processes related by (semi-typed) linear bisimilarity, which is implicit in the following technical development.

Since $\approx_{\mathrm{L}}$ is immediately reflexive and symmetric, it is an equivalence relation. We can also show $\equiv \subset \approx_{\mathrm{L}}$ by checking each equation. For example $(\boldsymbol{\nu}\, x)(P \mid Q) \approx_{\mathrm{L}} (\boldsymbol{\nu}\, x)P \mid Q$ with $x \notin \mathsf{fn}(Q)$, follows from Proposition 4 (4,5). For compatibility, closure under prefixes is easy. For restriction, by Lemma 1 (3), we can immediately show:

Proposition 7. (restriction) *Let* $P_1^{A_1} \approx_{\mathrm{L}}^{\Gamma \cdot x} P_2^{A_2}$ *with* $\mathsf{md}(A(x)) \in \{*, \mathbf{!}\}$. *Then* $(\boldsymbol{\nu}\, x)P^{A_1/x} \approx_{\mathrm{L}}^{\Gamma} (\boldsymbol{\nu}\, x)P_2^{A_2/x}$.

4.4 Congruency of $\approx_{\mathrm{L}}$ (2): Parallel Composition

Suppose we wish to prove the relation $\mathcal{R} \stackrel{\text{def}}{=} \{(P_1 \mid Q_1,\ P_2 \mid Q_2) \mid P_1 \approx P_2,\ Q_1 \approx Q_2\}$ to be a bisimulation up to restriction [27] in order to show that $\approx$ is closed under $\mid$. Assume $P_1 \mid Q_1 \stackrel{l}{\longrightarrow} P_1' \mid Q_1$; then by assumption, there exists $P_2 \stackrel{\hat{l}}{\Longrightarrow} P_2' \approx P_1'$, hence in the standard proof, we easily have: $P_2 \mid Q_2 \stackrel{\hat{l}}{\Longrightarrow} P_2' \mid Q_2$, and $P_1' \mid Q_1\ \mathcal{R}\ P_2' \mid Q_2$. However, due to strong invisibility, the same reasoning does not work for $\approx_{\mathrm{L}}$ even in the above trivial case. Recall the example in the Introduction, $P_1 \stackrel{\text{def}}{=} !u(x).\overline{x}\mathtt{in}_1$ and $P_2 \stackrel{\text{def}}{=} !u(x).\overline{y}(e)e.\overline{x}\mathtt{in}_1$. Then we know $P_1 \approx_{\mathrm{L}} P_2$ because $\overline{y}(e)$ and e are both invisible, so we have $P_1 \stackrel{u(x)}{\longrightarrow}_{uy} P_1' \stackrel{\text{def}}{=} \overline{x}\mathtt{in}_1 \mid P_1$ and $P_2 \stackrel{u(x)}{\Longrightarrow}_{uy} P_2' \stackrel{\text{def}}{=} \overline{x}\mathtt{in}_1 \mid P_2$ with P_1' and P_2' bisimilar. Suppose we compose them with $Q \stackrel{\text{def}}{=} !y(e).Q_0$ for some Q_0 such that $P_1 \mid Q$ and $P_2 \mid Q$ are typable. Then we have $P_1 \mid Q \stackrel{u(x)}{\longrightarrow}_{uy} P_1' \mid Q$, while we cannot have $P_2 \mid Q \stackrel{u(x)}{\Longrightarrow}_{uy} P_2' \mid Q$, because the only possible transition is $P_2 \mid Q \stackrel{u(x)}{\longrightarrow} \stackrel{\tau}{\longrightarrow} (\boldsymbol{\nu}\, e)(e.\overline{x}\mathtt{in}_1 \mid Q_0) \mid P_2 \mid Q$. In order to achieve $P_2' \stackrel{\text{def}}{=} \overline{x}\mathtt{in}_1 \mid P_2$ from this process, $e.\overline{x}\mathtt{in}_1$ needs an acknowledgement $\overline{e}$ from Q_0. Now we use a liveness property which extends Proposition 1 (3): if Q_0 has a linear output type at e, then there *always* exist a finite sequence of strong invisible transitions to emit e such that $Q_0 \stackrel{\overline{e}}{\Longrightarrow}_{uy} Q_0'$ and $Q \approx_{\mathrm{L}} Q_0' \mid Q$.

In the following we define such a chain, called *call-sequence*. Let us assume $P \stackrel{l_1 \cdot l_2}{\Longrightarrow} Q$. We write: $l_1 \frown_{\mathsf{b}} l_2$ (l_1 *binds* l_2) when the subject of l_1 is bound by l_2 (e.g. $x(y) \frown_{\mathsf{b}} \overline{y}$) and $l_1 \frown_{\mathsf{p}} l_2$ (l_1 *prefixes* l_2) when the action l_1 is input-prefixed by l_2 (e.g. $x(y) \frown_{\mathsf{p}} \overline{z}$ in $x(y).\overline{z}$). Define $\frown = \frown_{\mathsf{b}} \cup \frown_{\mathsf{p}}$. We write $\boldsymbol{\tau} \frown l_1$ if $P \stackrel{l_2}{\Longrightarrow} Q$ and P has subterms Q_1 and Q_2 such that $Q_1 \stackrel{l}{\Longrightarrow} Q_1'$ and $Q_2 \stackrel{\overline{l} \cdot l_2}{\Longrightarrow} Q_2'$ with $\overline{l} \frown l_2$; similarly we define $l_1 \frown \boldsymbol{\tau}$; we extend this to a chain $l_1 \frown \boldsymbol{\tau}^* \frown l_2$ and denote it $l_1 \frown^+ l_2$ ([7, Appendix F] gives a detailed definition using occurrences of terms). Then a *call-sequence (c.s.) to* l *under* A has the following shape.

$$(l_0 \frown^+)\ l_1 \frown_{\mathsf{b}} l_2 \frown^+ l_3 \frown_{\mathsf{b}} l_4 \frown^+ \cdots \frown^+ l_{2n-1} \frown_{\mathsf{b}} l_{2n} \frown_{\mathsf{p}} l$$

where $\mathsf{md}(l_{2k-1}^A) = \mathbf{?}$ and $\mathsf{md}(l_{2k}^A) = \downarrow$.

Lemma 2. (call-sequence) *Let* P *be typable below.*

1. (permutation) $P \stackrel{l}{\longrightarrow} \stackrel{l'}{\longrightarrow} P'$ *with* $l \not\frown l'$ *implies* $P \stackrel{l'}{\longrightarrow} \stackrel{l}{\longrightarrow} P'$.

2. (shortest c.s.) *Suppose* $P \stackrel{l}{\Longrightarrow}_\Gamma$ *with* l *output. Then there is a shortest c.s.* $l_1 \curvearrowright \cdots \curvearrowright l_n$ *to* l *such that* $P \stackrel{l_1 \cdots l_n}{\longrightarrow} \stackrel{l}{\longrightarrow}$.
3. (extended liveness) *Suppose* $\vdash P \triangleright A \otimes e : \tau$ *with* $\mathsf{md}(\tau) = \uparrow$. *Then* $P \Longrightarrow_{A \otimes e:\tau} \stackrel{l}{\longrightarrow}$ *with* $\mathsf{sbj}(l) = e$.
4. (branching-closed c.s) *Suppose* $P \stackrel{l}{\Longrightarrow}_\Gamma P'$ *with* l *linear output. Then there is* Δ*-branching-closed call sequences to* l *such that* $\{P \Longrightarrow_\Gamma P_k\}$ *with* $P_k \stackrel{l_k}{\longrightarrow} P'_k$ *and* $\mathsf{sbj}(l_k) = \mathsf{sbj}(l)$.

Proof. From the following, we sometimes write $\boldsymbol{\tau}$ as $(l,\ \overline{l})$ explicitly if $\boldsymbol{\tau}$ is done by interaction between (the indices of) l and $\overline{l}$.

(1) is obvious.

(2) By (1), we know there is a shortest transition sequence $P \stackrel{l_1 \cdots l_n}{\longrightarrow} \stackrel{l}{\longrightarrow}$ such that $l_1 \curvearrowright l_2 \cdots l_{n-1} \curvearrowright l_n \curvearrowright l$ and l_i is Γ-s.i. Suppose $\mathsf{md}(l^A) \in \{\mathbf{?}, \uparrow\}$. Then exactly one of the following must be true: (a) $\mathsf{md}(l_n) = \downarrow$ with $\mathsf{sbj}(l_n) \notin \Gamma$, or (b) $l_n = \boldsymbol{\tau}$, since if $\mathsf{md}(l_n) = \mathbf{!}$, then l_n is not Γ-s.i. and l_n should be input or $\boldsymbol{\tau}$.

Suppose case (a) holds. Then we have either (1) $l_n = l_1$ or (2) $l_{n-1} \curvearrowright_{\mathsf{b}} l_n$ with $\mathsf{md}(l_{n-1}) = \mathbf{?}$ since by the sequentiality constraint, linear outputs cannot directly carry linear inputs. We then set $n = n - 1$, and apply the same routine.

Now suppose case (b) holds. Then we have $\boldsymbol{\tau} \stackrel{\text{def}}{=} (l'_n,\ \overline{l_n}')$ with $\mathsf{md}(l'_n) \in \{\mathbf{?}, \uparrow\}$. If $l_{n-1} \curvearrowright l'_n$, then we just repeat the same routine by setting $n = n-1$. Suppose $l_{n-1} \curvearrowright \overline{l_n}$. By input typing rules, $l_{n-1} \curvearrowright_{\mathsf{b}} \overline{l_n}$. Then obviously $l_{n-1} = \boldsymbol{\tau} = (l'_{n-1},\ \overline{l_{n-1}}') \curvearrowright_{\mathsf{b}} l_n$ with $\overline{l_{n-1}}' \curvearrowright_{\mathsf{b}} l_n$. Hence we can use the same routine by setting $n = n - 1$. We repeat this procedure until we arrive at $n = 1$.

(3) First, given $\vdash P \triangleright A \otimes e : \tau$, there always exists a prime $\vdash R \triangleright B$ such that $B \asymp A$ and $A \odot B$ closed. Then by Proposition 1 (3), $P \,|\, R \twoheadrightarrow \stackrel{l}{\longrightarrow}$ with $\mathsf{sbj}(l) = e$. By (1), there is the shortest transitions $P \stackrel{l_1 \cdots l_n}{\longrightarrow} \stackrel{l}{\longrightarrow}$ such that $l_1 \curvearrowright l_2 \cdots l_{n-1} \curvearrowright l_n \curvearrowright l$. By the same reasoning as above, $l_1 \curvearrowright l_2 \cdots l_{n-1} \curvearrowright l_n$ is a call sequence, so all subjects of l_i with $\mathsf{md}(l_i) = \downarrow$ are bound by l_{i-1} except $i = 1$. Hence if $\mathsf{md}(l_1) = \mathbf{?}$, we are done. So suppose $\mathsf{md}(l_1) = \downarrow$. If $\mathsf{sbj}(l_1) = x \in \mathsf{fn}(P)$, then by definition of $\odot$, P should have a type $x : \tau' \to e : \tau$. This obviously contradicts the assumption P is typed by $A \otimes e : \tau$.

(4) By (2) and (3). ■

Note (3) does not restrict the shape of A, cf. Proposition 1 (3). Together with (2), we know there is always a shortest strongly invisible call sequence to each linear output.

We also use the following lemma. Below (1) is a basic up-to technique. (2, 3) say **?**-actions can be safely neglected. (4) essentially says $\approx_{\text{L}}$ is closed under injective renaming. (5,6) say that $\boldsymbol{\tau}$-actions, replicated inputs, linear outputs and unary linear inputs can be neglected up to $\approx_{\text{L}}$.

Lemma 3.

1. (up to restriction) *Let a semi-typed* $\mathcal{R}$ *satisfy that* $P_1^{A_1} \,\mathcal{R}\, P_2^{A_2}$ *with* $\Delta = \mathsf{fn}(A_1) \cap \mathsf{fn}(A_2)$ *implies the following and its symmetric case: whenever* $P_1^{A_1} \stackrel{l}{\longrightarrow}$

$(\boldsymbol{\nu}\,\vec{x})Q_1^{B_1/\vec{x}}$, *there is a* Δ*-b.c. closed* $\{P_2^{A_2} \stackrel{\hat{l}}{\Longrightarrow}_\Delta (\boldsymbol{\nu}\,\vec{x})Q_{2i}^{B_{2i}/\vec{x}}\}_{i\in I}$ *such that* $Q_1^{B_1}\,\mathcal{R}\,Q_{2i}^{B_{2i}}$ *for all* $i \in I$*. Then* $\mathcal{R}$ *is a bisimulation.*

2. (innocuousness of **?**) *Suppose* $\vdash P|R_i \triangleright A\otimes B_i$ *where* $\vdash P\triangleright A$ *and* $\vdash R_i\triangleright ?A_i\otimes B_i$ *with* $A_i \asymp A$ *(*$i = 1, 2$*) and* $\mathsf{fn}(B_1)\cap\mathsf{fn}(B_2) = \emptyset$*. Then* $(P \mid R_1)^{A\otimes B_1} \approx_{\mathrm{L}} (P \mid R_2)^{A\otimes B_2}$*.*
3. (renaming on **?**) *Suppose* $\vdash P\triangleright ?A \odot B$ *and* σ *is an injective renaming such that* $\mathsf{dom}(\sigma) = \mathsf{fn}(A)$ *and* $\mathsf{cod}(\sigma)\cap\mathsf{fn}(P) = \emptyset$*. Then* $P^{A\odot B} \approx_{\mathrm{L}}^{\mathsf{fn}(B)} P\sigma^{A\sigma\odot B}$*.*
4. (renaming) *Suppose* $Q \approx_{\mathrm{L}}^{\Delta} P$ *and* σ *is an injective renaming such that* (a) $\mathsf{dom}(\sigma)\cap\Delta = \emptyset$*, and* (b) $\mathsf{cod}(\sigma)\cap(\mathsf{fn}(P)\cup\mathsf{fn}(Q)) = \emptyset$*. Then* $Q \approx_{\mathrm{L}}^{\Delta} P\sigma$*.*
5. (negligible actions) *Suppose* $\vdash P \triangleright A$ *and* $P \stackrel{l}{\longrightarrow} P'$ *with* $l = \boldsymbol{\tau}$ *or* $\mathsf{md}(l^A) \in \{\mathbf{!}, \uparrow\}$*. Then* $P \approx_{\mathrm{L}} P'$*.*
6. (linear unary input) *Suppose* $\vdash P\triangleright A$ *and* $P \stackrel{x(\vec{y})}{\longrightarrow} P'$ *with* $\mathsf{md}(A(x)) = \downarrow$*. Then* $P \approx_{\mathrm{L}}^{\Gamma} P'$*.*

Proof. See Appendix F.1. ■

In the following lemma, (5) says that if a process moves via a subject which is not in Δ, it does not change its meaning at Δ; (7) says that a Δ-closed call sequences in which every branch gives the same answer (i.e. linear output) can be regarded as a single thread of the standard weak transition, i.e. $\Longrightarrow$.

Lemma 4.

1. *Let* $\vdash P \triangleright A$*. Assume* $P \approx_{\mathrm{L}}^{\Delta} Q$ *and* $P \stackrel{l}{\longrightarrow} P'$ *where* l *satisfies either* (1) $l = \boldsymbol{\tau}$*,* (2) $\mathsf{md}(l^A) \in \{\mathbf{!}, \uparrow\}$ *with* $\mathsf{sbj}(l) \notin \Delta$*, or* (3) $l = x(\vec{y})$ *and* $\mathsf{md}(l^A) = \downarrow$ *with* $\mathsf{sbj}(l) \notin \Delta$*. Then* $P' \approx_{\mathrm{L}}^{\Delta} Q$*.*
2. *Suppose* $\{P \Longrightarrow_\Delta P_j\}$ *is* Δ*-b.c and* $P_j \approx_{\mathrm{L}}^{\Delta} Q$ *for all* j *with* $\Delta = \mathsf{fn}(P)\cap\mathsf{fn}(Q)$*. Then* $P \approx_{\mathrm{L}}^{\Delta} Q$*.*
3. *Suppose* $P \approx_{\mathrm{L}}^{\Delta} Q$ *and* $P \stackrel{l}{\longrightarrow} P'$ *with* $l = x\mathtt{in}_i(\vec{y})$ *with* $\mathsf{sbj}(l) \notin \Delta$*. Then* $P' \approx_{\mathrm{L}}^{\Delta} Q$*.*
4. *Suppose* $P \approx_{\mathrm{L}}^{\Delta} Q$ *and* $P \stackrel{l}{\longrightarrow} P'$ *with* $\mathsf{md}(l) = \mathbf{?}$ *with* $\mathsf{sbj}(l) \notin \Delta$*. Then* $P' \approx_{\mathrm{L}}^{\Delta} Q$*.*
5. *Suppose* $P \approx_{\mathrm{L}}^{\Delta} Q$ *and* $P \stackrel{l}{\longrightarrow} P'$ *where* l *is* Δ*-invisible with* $\mathsf{sbj}(l) \notin \Delta$*. Then* $P' \approx_{\mathrm{L}}^{\Delta} Q$*.*
6. *Suppose* $P \approx_{\mathrm{L}}^{\Delta} Q$*. Assume* $P \stackrel{l_1\cdots l_n}{\Longrightarrow} \stackrel{l}{\longrightarrow} P'$ *where* $l_1\cdots l_n$ *is a* Δ*-strong invisible call sequence to* l *such that* $\mathsf{sbj}(l) \notin \Delta$*. Then* $P' \approx_{\mathrm{L}}^{\Delta} Q$*.*
7. *Suppose* $P \approx_{\mathrm{L}}^{\Delta} Q$*. Assume* $\{P \Longrightarrow_\Delta P_j\}$ *is* Δ*-branching closed call sequences, and for all* j*, we have* $P_j \stackrel{l}{\longrightarrow} P_j'$ *with* l *linear output. Then* $Q \approx_{\mathrm{L}}^{\Delta} P_j$*.*

Proof. See Appendix F.2. ■

Using Lemma 4 (2), we can reduce the task of checking a bisimulation closure in linear processes.

Lemma 5. *Suppose $\mathcal{R}$ is a semi-typed relation such that when $P_1^{A_1}\,\mathcal{R}\,P_2^{A_2}$ with $\Delta = \mathsf{fn}(A_1)\cap\mathsf{fn}(A_2)$ implies the following and its symmetric case: whenever $P_1^{A_1}\xrightarrow{l}Q_1^{B_1}$, there is a Δ-closed $\{P_2^{A_2}\Longrightarrow_\Delta\xrightarrow{\hat{l}}_\Delta Q_{2i}^{B_{2i}}\}_{i\in I}$ such that $Q_1^{B_1}\,\mathcal{R}\,Q_{2i}^{B_{2i}}$ for all $i\in I$. Call such a relation a* restricted bisimulation. *Then the maximum restricted bisimulation coincides with $\approx_{\text{L}}$.*

Proof. Let $\overset{\circ}{\approx}_{\text{L}}$ be the maximum restricted bisimulation. First, obviously $\overset{\circ}{\approx}_{\text{L}}\subseteq\approx_{\text{L}}$. For the reverse inclusion we show that $\approx_{\text{L}}$ is a restricted bisimulation. Let us assume $P_1\approx_{\text{L}}P_2$ and $P_1^{A_1}\xrightarrow{l}Q_1^{B_1}$. Then there is a Δ-closed $\{P_2^{A_2}\overset{\hat{l}}{\Longrightarrow}_\Delta Q_{2i}^{B_{2i}}\}_{i\in I}$ such that $Q_1^{B_1}\approx_{\text{L}}Q_{2i}^{B_{2i}}$ for all $i\in I$. First, we can write $P_2^{A_2}\Longrightarrow_\Delta\xrightarrow{\hat{l}}_\Delta Q_{2ki}^{B_{2ki}}\Longrightarrow_\Delta Q_{2i}^{B_{2i}}$ by definition. Then by the Split Lemma (Lemma 1 (2)), $\{Q_{2k_i}^{B_{2k_i}}\Longrightarrow_\Delta Q_{2i}^{B_{2i}}\}$ is Δ-b.c. for each k_i. Then by Lemma 4 (2), $Q_1^{B_1}\approx_{\text{L}}Q_{2i}^{B_{2i}}$ implies $Q_1^{B_1}\approx_{\text{L}}Q_{2k_i}^{B_{2k_i}}$ for each k_i. Hence $\approx_{\text{L}}$ is a restricted bisimulation, that is $\approx_{\text{L}}\subseteq\overset{\circ}{\approx}_{\text{L}}$, as required. ■

By the following result, we can further reduce the conditions needed for a bisimulation closure. The reduction comes from Proposition 4 (3)) (input availability) and Lemma 4 (5). The form of the resulting relation, as stated in the following proposition, is similar to the branching bisimulation studied in (untyped) confluence processes [28].

Lemma 6. *Suppose $\mathcal{R}$ is a semi-typed relation such that when $P_1^{A_1}\,\mathcal{R}\,P_2^{A_2}$ with $\Delta=\mathsf{fn}(A_1)\cap\mathsf{fn}(A_2)$ implies the following and its symmetric case:*

- *whenever $P_1^{A_1}\xrightarrow{l}Q_1^{B_1}$ where l is Δ-invisible and $\mathsf{sbj}(l)\notin\Delta$, then $P_2^{A_2}\,\mathcal{R}\,Q_1^{B_1}$.*
- *whenever $P_1^{A_1}\xrightarrow{l}Q_1^{B_1}$ with l input such that $\mathsf{sbj}(l)\in\Delta$, then there is $P_2^{A_2}\xrightarrow{l}_\Delta Q_2^{B_2}$ such that $Q_1^{B_1}\,\mathcal{R}\,Q_2^{B_2}$.*
- *whenever $P_1^{A_1}\xrightarrow{l}Q_1^{B_1}$ with l Δ-visible linear output such that $\mathsf{sbj}(l)\in\Delta$, then there is a Δ-closed call sequences to l $\{P_2^{A_2}\Longrightarrow_\Delta\xrightarrow{l}Q_{2i}^{B_{2i}}\}_{i\in I}$ such that $Q_1^{B_1}\,\mathcal{R}\,Q_{2i}^{B_{2i}}$ for all $i\in I$.*
- *whenever $P_1^{A_1}\xrightarrow{l}Q_1^{B_1}$ with $\mathsf{md}(l^{A_1})=\mathbf{?}$ and $\mathsf{sbj}(l)\in\Delta$, there is a Δ-closed call sequence to l, $\{P_2^{A_2}\Longrightarrow_\Delta Q_{2i}^{B_{2i}}\}_{i\in I}$, such that either $Q_1^{B_1}\,\mathcal{R}\,Q_{2i}^{B_{2i}}$ or $Q_{2i}^{B_{2i}}\xrightarrow{l}Q'^{B'_{2i}}_{2i}$ such that $Q_1^{B_1}\,\mathcal{R}\,Q'^{B'_{2i}}_{2i}$.*

Then the maximum such relation, denoted by $\overset{\bullet}{\approx}_{\text{L}}$, coincides with $\approx_{\text{L}}$.

Proof. By Lemma 5, we shall show $\overset{\bullet}{\approx}_{\text{L}}$ coincides with $\overset{\circ}{\approx}_{\text{L}}$. Obviously $\overset{\bullet}{\approx}_{\text{L}}\subseteq\overset{\circ}{\approx}_{\text{L}}$. Hence we show $\overset{\circ}{\approx}_{\text{L}}\subseteq\overset{\bullet}{\approx}_{\text{L}}$ by showing $\overset{\circ}{\approx}_{\text{L}}$ is a (further) restricted form of bisimulation given in Lemma 6. Assume $P_1\overset{\circ}{\approx}_{\text{L}}P_2$.

Case 1: Suppose $P_1\xrightarrow{l}Q_1$ where l is $\boldsymbol{\tau}$ or l is Δ-invisible and $\mathsf{sbj}(l)\notin\Delta$. By Lemma 4 (1,5), we always have $P_2\overset{\circ}{\approx}_{\text{L}}Q_1$. Hence by $P_2\overset{\varepsilon}{\Longrightarrow}P_2$, we have $P_2\overset{\circ}{\approx}_{\text{L}}Q_1$.

Case 2: Suppose $P_1 \xrightarrow{l} Q_1$ with $\mathsf{sbj}(l) \in \Delta$ and l input, and there is a Δ-b.c. $\{P_2^{A_2} \stackrel{l}{\Longrightarrow}_\Delta Q_{2i}^{B_{2i}}\}_{i\in I}$ such that $Q_1^{B_1} \mathrel{\mathring{\approx}}_{\mathrm{L}} Q_{2i}^{B_{2i}}$. Then by (permutation) and (input availability), we always have $P_2^{A_2} \xrightarrow{l} P_3^{A_3} \Longrightarrow_\Delta Q_{2i}^{B_{2i}}$. Then by Lemma 4 (2), $P_3^{A_3} \mathrel{\mathring{\approx}}_{\mathrm{L}} P_2^{A_2}$ as in the proof of Lemma 5.

Case 3: Suppose $P_1 \xrightarrow{l} P_2$ with $\mathsf{sbj}(l) \in \Delta$ and l linear output, and there is a Δ-b.c. $\{P_2^{A_2} \stackrel{l}{\Longrightarrow}_\Delta Q_{2i}^{B_{2i}}\}_{i\in I}$ such that $Q_1^{B_1} \approx_{\mathrm{L}} Q_{2i}^{B_{2i}}$. Then by (permutation) and Lemma 2 (2, shortest c.s.), we always have a Δ-closed call-sequence $\{P_2^{A_2} \Longrightarrow_\Delta R_{2j}^{C_{2j}}\}_{j\in J}$ such that $R_{2j} \xrightarrow{l} R'_{2j} \Longrightarrow_\Delta Q_{2i}^{B_{2i}}$. Then we use Proposition 4 (2) as in the above cases in order to obtain $Q_1 \mathrel{\mathring{\approx}}_{\mathrm{L}} R'_{2j}$.

Case 4: Suppose $P_1 \xrightarrow{l} P_2$ with l a replicated output. This case is the same as Case 3 above by using (permutation), Lemma 2 (2) and Proposition 4 (2).

This exhaust all cases, hence we know $\mathring{\approx}_{\mathrm{L}} \subseteq \overset{\bullet}{\approx}_{\mathrm{L}}$.

Lemma 7. (composition) *Suppose* $\vdash P \triangleright A$ *and* $\vdash Q_i \triangleright B_i$ *with* $Q_1 \approx^\Delta Q_2$, $A \asymp B_i$ *and* $\mathsf{fn}(A) \cap \mathsf{fn}(B_1) \supseteq \mathsf{fn}(A) \cap \mathsf{fn}(B_2) = \Delta$. *Moreover the linear output outside of* Δ *is not suppressed by a linear input at* Δ. *Assume* $\{P \Longrightarrow_\Delta P_k\}$ *is* Δ*-b.c. Then there exists a* Δ*-b.c.* $\{(P \mid Q_1) \Longrightarrow_\Delta (\nu\vec{w}_j)(P'_j \mid Q_j)\}$ *such that* $Q_j \approx_{\mathrm{L}}^\Delta Q_2$ *and* $P'_j \approx_{\mathrm{L}} P_k$ *for some* k.

Proof. See Appendix F.3. ■

We are ready to prove the closure under parallel composition.

Proposition 8. (parallel composition) *Let* $P_1^A \approx_{\mathrm{L}} P_2^A$ *and* $Q_1^B \approx_{\mathrm{L}} Q_2^B$ *such that* $A \asymp B$. *Then* $P_1 | Q_1^{A\odot B} \approx_{\mathrm{L}} P_2 | Q_2^{A\odot B}$.

Proof. We use the characterisation of $\approx_{\mathrm{L}}$ in Lemma 6. Let $\mathcal{R}$ be generated by: $(\boldsymbol{\nu}\,\vec{w}_1)(P_1^{A_1} \mid Q_1^{B_1})\ \mathcal{R}\ (\boldsymbol{\nu}\,\vec{w}_2)(P_2^{A_2} \mid Q_2^{B_2})$ iff $P_1 \approx_{\mathrm{L}}^{\Delta_1} P_2$, $Q_1 \approx_{\mathrm{L}}^{\Delta_2} Q_2$, $\Delta = \Delta_1 \cup \Delta_2$, $(\Delta \setminus \Delta_i) \cap (\mathsf{fn}(P_i) \cup \mathsf{fn}(Q_i)) = \emptyset$, $\{\vec{w}_i\} = (\mathsf{fn}(P_i) \cap \mathsf{fn}(Q_i)) \setminus \Delta$ such that any linear output outside of Δ is not suppressed by a linear input at Δ. We show $\mathcal{R}$ is a bisimulation. First suppose $(\boldsymbol{\nu}\,\vec{w}_1)(P_1 \mid Q_1) \xrightarrow{l} (\boldsymbol{\nu}\,\vec{w}_1)(P'_1 \mid Q_1)$ with $P_1 \xrightarrow{l} P'_1$.

Case 1: Suppose $\mathsf{sbj}(l) \in \Delta$ and l is an input. Then by assumption, we have $P_2 \xrightarrow{l}_\Delta P'_2 \approx_{\mathrm{L}} P'_1$. Also obviously $x : \tau \in |A_2 \odot B_2|$ with $\mathsf{md}(\tau) \in \{\downarrow, \mathbf{!}\}$. Hence by (input availability), $(\boldsymbol{\nu}\,\vec{w}_2)(P_2 \mid Q_2) \xrightarrow{l} (\boldsymbol{\nu}\,\vec{w}_2)(P'_2 \mid Q_2)$, as required.

Case 2: Suppose $\mathsf{sbj}(l) \notin \Delta$ and l is invisible. Then by assumption, we have $P_2 \approx_{\mathrm{L}} P'_1$. Hence we have $(\boldsymbol{\nu}\,\vec{w}_2)(P_2 \mid Q_2) \stackrel{\varepsilon}{\Longrightarrow} (\boldsymbol{\nu}\,\vec{w}_2)(P_2 \mid Q_2)\ \mathcal{R}\ (\boldsymbol{\nu}\,\vec{w}_1)(P'_1 \mid Q_1)$.

Case 3: Suppose $\mathsf{sbj}(l) \in \Delta$ and l is a Δ-visible linear output. Then by assumption, there exits a Δ-branching closed call sequence $\{P_2 \Longrightarrow_\Delta P'_{2i}\}$ such that $P'_{2i} \xrightarrow{l} P'_{3i} \approx_{\mathrm{L}} P'_1$. Then by Lemma 7, we have a Δ-branching closed call sequences $\{(\boldsymbol{\nu}\,\vec{w}_2)(P_2 \mid Q_2) \Longrightarrow_\Delta (\boldsymbol{\nu}\,\vec{w}_2{}')(P''_{2k} \mid Q_{2k})\}$ such that $Q_{2k} \approx_{\mathrm{L}} Q_1$, $P''_{2k} \approx_{\mathrm{L}} P''_{2i}$, and $P''_{2k} \xrightarrow{l} P''_{3k}$. By Lemma 4 (2), $P''_{3k} \approx_{\mathrm{L}} P''_{3i}$. Hence noting

$\mathsf{fn}(P''_{3k}) \cap \mathsf{fn}(P_{3i}) \cap \mathsf{fn}(P'_1) = \mathsf{fn}(P''_{3k}) \cap \mathsf{fn}(P'_1)$, we use transitivity to obtain $P''_{3k} \approx_{\mathrm{L}} P'_1$. Hence $(\boldsymbol{\nu}\, \vec{w}_2{}')(P''_{3k} \mid Q_{2k})\ \mathcal{R}\ (\boldsymbol{\nu}\, \vec{w}_1)(P'_1 \mid Q_1)$, as required.

Case 4: Suppose l is a replicated output. This case is similar to the above using Lemma 7 again.

For the condition of suppression, we check that it is maintained during any transition $P \xrightarrow{l} P'$ in general. This is straightforward by the typing rules. For example, suppose $P^{A \otimes x:\tau} \xrightarrow{l} P'^{A \otimes y:\tau' \to x:\tau}$ with $\mathsf{md}(\tau') = \downarrow$. By $y \in \mathsf{bn}(l)$, l is a replicated output, and obviously $y \notin \mathsf{fn}(P)$. Hence if $x \notin \Delta$ is not suppressed by the input at Δ from the beginning as in the assumption of $\mathcal{R}$, it is maintained during any transitions. Hence we are done.

Case 5: If $l = \boldsymbol{\tau}$ then $P'_1 \approx_{\mathrm{L}} P_1$ by Lemma 3 (5). Hence by transitivity the inaction can simulate this as in Case 2.

Second, if only Q_1 has an action, this is the same as P_1. Finally the case when $P_1|Q_1 \xrightarrow{\boldsymbol{\tau}} (\boldsymbol{\nu}\, \vec{w})(P'_1|P'_2)$ from $P_1 \xrightarrow{l} P'_1$ and $Q_1 \xrightarrow{\overline{l}} Q'_1$ is the standard reasoning except that we apply the same technique as above for processing additional invisible actions. Assume, without loss of generality, that l is output. If l is at Δ and this action is not invisible, then (since its dual input is always possible) we can reason precisely as above. If l is done at Δ but it is invisible, then it is possible that a subset of the simulating b.c. set abstract away l. In this case, however, the dual action is negligible by Lemma 3 (5) and (6), which gives us the required closure. If l is not at Δ then the closure is immediate using Lemma 6. This exhausts all cases. ■

Thus we conclude the centre of $\approx_{\mathrm{L}}$ is a congruence. Since the observability predicate given in § 2.3 is easily satisfied by $\approx_{\mathrm{L}}$, we also know $\approx_{\mathrm{L}} \subseteq \cong_{\pi}$.

Remark 3. The basic framework of the above proof for closure under parallel composition extends to the settings where we combine such elements as state, nontermination and nondeterminism with linearity (though certain simplification in the proof above is possible due to the pure linear structure, in particular the use of reduced conditions for bisimulation, given in Lemmas 6 and 5). The essential properties we need for this is those of call-sequences, as given in Lemma 2. As examples of such combination, the reader may refer to [23]. See also Remark 4 at the end of Section 5.

5 Applications to Secrecy

In linear bisimilarity, we abstract away non-affecting typed actions as if they were $\boldsymbol{\tau}$-actions. If we assign a secrecy level to each channel and stipulate a level of observation, then we can further abstract away those actions which should not be visible from the stipulated level. For example, from a low-level viewpoint, actions at high-level channels should be invisible. The technical development of this secrecy enhancement closely follows that of the linear bisimilarity, and offers a powerful tool for reasoning about secrecy in processes.

Assume given a complete lattice of *secrecy levels* $(s, s', \ldots)$ with the ordering $\sqsubseteq$. $\top$ (the most secret) and $\bot$ (the most public) denote the top and bottom of the lattice, respectively. Channel types are annotated with these levels:

$$\tau \ ::= \ \tau_{\mathtt{I}} \mid \tau_{\mathtt{O}} \mid *_s \qquad \tau_{\mathtt{I}} \ ::= \ (\vec{\tau}_{\mathtt{O}})^{p_{\mathrm{I}}}_s \mid [\&_{i\in I}\vec{\tau}_{\mathtt{O}i}]^{p_{\mathrm{I}}}_s \qquad \tau_{\mathtt{O}} \ ::= \ (\vec{\tau}_{\mathtt{I}})^{p_{\mathrm{O}}}_s \mid [\oplus_{i\in I}\vec{\tau}_{\mathtt{I}i}]^{p_{\mathrm{O}}}_s$$

The same constraints as before apply to channel types. In $\overline{\tau}$, we require each dualised occurrence to own identical secrecy levels. Action types are given precisely as before, using secrecy annotated types. $\vdash P \triangleright A$ (or P^A) is derived by the same rules as in Appendix B (the secrecy annotations on types do not affect the derivation of typing judgements). We set:

1. l^A is *s-affecting* if it is affecting in the preceding sense and, if l is a linear selection, then $\mathsf{sec}(A(x)) \sqsubseteq s$ ($\mathsf{sec}(\tau)$ is the outermost secrecy level of τ).
2. l^A is *s-Δ-invisible* when either $\mathsf{fn}(l)\cap\Delta = \emptyset$ or, if not, l^A is an output which is not s-affecting. If l^A is s-Δ-invisible and, moreover, is enabled, then l^A is *s-Δ strongly invisible*. The abstracted transitions $P^A \Longrightarrow_{\Delta,s} Q^B$, $P^A \stackrel{l}{\Longrightarrow}_{\Delta,s} Q^B$ and $P^A \stackrel{\hat{l}}{\Longrightarrow}_{\Delta,s} Q^B$ and the associated underlying transitions are defined accordingly.

In (1) we only count linear selections because in the linear type structure no other types directly emit information. We can now introduce the bisimulation.

Definition 7. (s-bisimulation) A semi-typed relation $\mathcal{R}$ is a *s-bisimulation* when $P_1^{A_1}\,\mathcal{R}\,P_2^{A_2}$ with $\Delta = \mathsf{fn}(A_1)\cap\mathsf{fn}(A_2)$ implies the following and its symmetric case: whenever $P_1^{A_1} \stackrel{l}{\longrightarrow} Q_1^{B_1}$, there is a Δ-closed $\{P_2^{A_2} \stackrel{\hat{l}}{\Longrightarrow}_{\Delta,s} Q_{2i}^{B_{2i}}\}$ such that $Q_1^{B_1}\,\mathcal{R}\,Q_{2i}^{B_{2i}}$. The maximum s-bisimulation exists for each s, which we write $\approx_s$.

By definition, $P^A \approx_{\mathrm{L}} Q^B$ implies $P^A \approx_s Q^B$ for any s. Further $P^A \approx_\top Q^B$ implies $P^A \approx_{\mathrm{L}} Q^B$. A simple example of s-bisimilarity:

Example 5. (s-bisimilarity) $\vdash \overline{x}\mathtt{in}_1 \triangleright x : \mathbb{B}_\top \not\approx_\top \vdash \overline{x}\mathtt{in}_2 \triangleright x : \mathbb{B}_\top$ but we have $\vdash \overline{x}\mathtt{in}_1 \triangleright x : \mathbb{B}_\top \approx_\bot \vdash \overline{x}\mathtt{in}_2 \triangleright x : \mathbb{B}_\top$.

A basic observation on $\approx_s$ is that it alone does not form a coherent notion of process equivalence.

Fact 1. *The centre of $\approx_s$ is not closed under parallel composition.*

Proof. Take $\overline{x}\mathtt{in}_i^{x:\tau_1}$ $(i = 1, 2)$ with $\tau_1 = \mathbb{B}_\top$. Then $\overline{x}\mathtt{in}_1 \approx_\bot \overline{x}\mathtt{in}_2$. However if we compose these processes with $x[.\overline{u}\mathtt{in}_1 \& .\overline{u}\mathtt{in}_2]^{x:\tau_2\to u:\tau_3}$ where $\tau_2 = \overline{\tau_1}$ and $\tau_3 = \mathbb{B}_\bot$, then $(\boldsymbol{\nu}\, x)(P_1|Q)^{u:\tau_3} \not\approx_\bot (\boldsymbol{\nu}\, x)(P_2|Q)^{u:\tau_3}$. ■

The example in the proof above suggests that, for regaining compositionality in $\approx_s$, we need to restrict the set of processes to those which do not transfer information at some high-level to lower levels. In other words, we require information flow in processes to be *secure*. Below we say l^A is *receiving at s* if l^A is a linear branching and moreover $\mathsf{sec}(A(\mathsf{sbj}(l))) = s$.

Definition 8. (behavioural secrecy) A set of typed processes $\mathcal{S}$ is a *secrecy witness* if the following holds: whenever $P^A \in \mathcal{S}$ and $P^A \xrightarrow{l} Q^B$, we have (1) $Q^B \in \mathcal{S}$ and (2) if l^A is receiving at s then $P^A \approx_{s'} Q^B$ for each s' such that $s \not\sqsubseteq s'$. P^A is *behaviourally secure* iff P^A is in some secrecy witness.

Only linear branching counts as "receiving", which is an exact dual of $\approx_s$ (where we consider abstraction by secrecy levels only for linear selection). Intuitively, a process is behaviourally secure iff, whenever it receives non-trivial information at some level, it behaves, to a lower-level observer, as if the action had not taken place. Some examples of (non-)secure processes follow.

Example 6. $\mathbf{0}^{\emptyset}$ is secure. If P^A is secure and $(\boldsymbol{\nu}\, x)P^{A/x}$ is well-typed, the latter is secure. If $P^{\vec{y}:\vec{\tau}\otimes ?\, A}$ is secure and $!x(\vec{y}).P^{x:(\vec{\tau})^{!}\to A}$ is well-typed, the latter is secure. Finally, given $A \stackrel{\text{def}}{=} x : \mathbb{B}_{\top} \to y : \overline{\mathbb{B}}_{\bot}$, $x[.\overline{y}\mathtt{in}_1 \& .\overline{y}\mathtt{in}_2]^A$ is not secure but $x[.\overline{y}\mathtt{in}_1 \& .\overline{y}\mathtt{in}_1]^A$ is secure (the latter is because $x[.\overline{y}\mathtt{in}_1 \& .\overline{y}\mathtt{in}_1]^{\,A} \approx^{y}_{\bot} \overline{y}\mathtt{in}_1^{A/x}$).

The following is proved precisely as Theorem 1 except that the use of s-invisibility is compensated by behavioural secrecy.

Proposition 9. *The centre of $\approx_s$ over behaviourally secure processes is compatible with all operators except linear branching.*

Proof. The essence of the proof is the same as that of Proposition 8, using the corresponding properties. The only difference is in the following points: assume, as in the proof of Proposition 8, that $(\boldsymbol{\nu}\, \vec{w}_1)(P_2|Q_2)$ simulates an action by $(\boldsymbol{\nu}\, \vec{w}_1)(P_1|Q_1)$.

1. When only P_1 has an action, we simply replace all statements "Δ-invisible" and "Δ-visible" with "s-Δ-invisible" and "s-Δ-visible", respectively. Using the corresponding lemmas, the same reasoning gives the required results.
2. When P_1 and Q_1 interact, it is possible that, say, P_1 has a linear selection which can be neglected at level s. Then P_2 may not simulate this action, but it is compensated by Q_1 not changing its semantics by the receiving action up to $\approx_{\mathrm{L}}$: thus $P_2|Q_2$ can simulate this action by the inaction, resulting in the same closure.

The second case crucially uses behavioural secrecy for composability. ■

Combined with a secure version of the linear branching rule,[5] Proposition 9 offers a framework for fully compositional reasoning about secrecy in linear processes. Via embeddings, it can be used for analysing secrecy for λ-terms [3] and, with extensions to the type structure, for sequential and concurrent imperative programs [8, 31, 32].

To investigate the relationship with the type-based approach in [23], we introduce $\mathsf{tamp}(A)$ (the lowest possible effect level of A) and $\cong_s$ (a secrecy-sensitive contextual congruence), both from [23].

[5] Given $\vdash P_i \triangleright C_i$ where $C_i = \vec{y}_i : \vec{\tau}_i \otimes \uparrow A^{-x} \otimes ?\, B^{-x}$ $(i = 1, 2)$, the rule requires that, in the antecedents, $P_i^{C_i} \approx_{s'} P_j^{C_j}$ for any s' such that $s \not\sqsubseteq s'$, in order to conclude that $\vdash x[\&_i(\vec{y}_i).P_i] \triangleright (x : [\&_i \vec{\tau}_i]^{\downarrow}_{s} \to A) \otimes B$.

Definition 9. (tamper level, [23]) The tamper level of τ, denoted $\mathsf{tamp}(\tau)$, is defined as follows. We assume $\vec{\tau} = \tau_1..\tau_i..\tau_n$ and $\vec{\tau}_i = \tau_{i1}..\tau_{ij}..\tau_{in_i}$.

1. $\mathsf{tamp}(\tau) = \top$ if τ is not affecting.
2. $\mathsf{tamp}((\vec{\tau})^{\uparrow}_s) = \sqcap_i \mathsf{tamp}(\tau_i)$, $\mathsf{tamp}([\oplus_i \vec{\tau}_i]^{\uparrow}_s) \stackrel{\text{def}}{=} s$, $\mathsf{tamp}((\vec{\tau})^{\mathbf{!}}_s) \stackrel{\text{def}}{=} \sqcap_i \mathsf{tamp}(\tau_i)$, $\mathsf{tamp}([\&_i \vec{\tau}_i]^{\mathbf{!}}_s) \stackrel{\text{def}}{=} \sqcap_{ij} \mathsf{tamp}(\tau_{ij})$.

Then we set $\mathsf{tamp}(A) \stackrel{\text{def}}{=} \sqcap\{\mathsf{tamp}(\tau) \mid x : \tau \text{ occurs in } A \text{ for some } x\}$.

Observe $\mathsf{tamp}(\tau) = \top$ whenever τ is not affecting.

Definition 10. (secrecy-sensitive contextual equality) For each s, s-contextual congruence $\cong_s$ is defined as the maximum typed congruence satisfying the following condition.

$$\text{If } P \Downarrow^i_x \text{ and } P \cong^{x:\mathbb{B}_{s'}}_{\pi} Q \text{ with } s' \sqsubseteq s, \text{ then } Q \Downarrow^i_x \qquad (i = 1, 2)$$

Proposition 10. *All secrecy typing rules in Appendix E are valid when typability is replaced with behavioural secrecy. Securely typed linear processes are a proper subset of behaviourally secure linear processes.*

Proof. By induction on the derivation of typing judgements in the secure typing system of Appendix E. The only non-immediate case is $(\mathsf{Bra}^{\downarrow})$. Let the behavioural security of P_i be witnessed by $S_i (i \in I)$. Define

$$S = \{x[\&_i(\vec{y_i})P_i]\} \cup \bigcup_{i \in I} S_i.$$

To see that S is a secrecy witness it is clearly sufficient to show that $\mathcal{R} \cup \approx_s$ is an s-bisimulation where $\mathcal{R}$ is given by: $x[\&_i(\vec{y_i})P_i]^A \ \mathcal{R} \ P_i^B$ for all $i \in I$. Clearly $\mathcal{R} \cup \approx_s$ is semi-typed. Let $\Delta = \mathsf{fn}(A) \cap \mathsf{fn}(B)$. If

$$x[\&_i(\vec{y_i})P_i] \xrightarrow{l} P_j \qquad \text{where } l = x\mathtt{in}_j(\vec{y_j}),$$

$\mathsf{fn}(l) \cap \Delta = \emptyset$, because $x \notin \mathsf{fn}(B)$. So l is s-Δ strongly invisible. This means that $P_j^B \xrightarrow{\hat{l}} P_j^B$ and hence $P_j^B \Longrightarrow_{\Delta,s} P_j^B$ for all $j \in I$. But $\{P_j^B \Longrightarrow_{\Delta,s} P_j^B\}_{j \in I}$ is Δ-branching closed and, by assumptions, $P_j \approx_s P_k$ for all $j, k \in I$, so we have indeed found a matching set of transitions.

Conversely, if $P_i^{A_i} \xrightarrow{l'} Q^B$ then $x[\&_i(\vec{y_i})P_i] \Longrightarrow_{\Delta,s} P_j$ for all $j \in I$ by the s-Δ strong invisibility of l. As $P_j \approx_s P_i$, we can find Δ-branching closed sets $\{P_j \Longrightarrow_{\Delta,s} Q_{jk}\}_{j,k \in I}$. Using Lemma 1(1) $\{x[\&_i(\vec{y_i})P_i] \Longrightarrow_{\Delta,s} P_j \Longrightarrow_{\Delta,s} Q_{jk}\}$ is a matching Δ branching closed set of transitions.

As an illustration of how to deal with the remaining rules, we now consider (Par). Assume $\vdash P_i \triangleright A_i \in S_i (i = 1, 2)$ and define

$$S = \{(\boldsymbol{\nu}\, \vec{x})(Q_1|Q_2)^B \mid Q_i^{B_i} \in S_i\}.$$

Assume that $(\boldsymbol{\nu}\,\vec{x})(Q_1|Q_2) \xrightarrow{l} R$. Induction on the derivation of that transition has essentially two cases. The first is this one.

$$\dfrac{\dfrac{\dfrac{Q_1 \xrightarrow{l} Q_1'}{Q_1|Q_2 \xrightarrow{l} Q_1'|Q_2}}{\vdots}}{(\boldsymbol{\nu}\,\vec{x})(Q_1|Q_2) \xrightarrow{l} (\boldsymbol{\nu}\,\vec{x})(Q_1'|Q_2)}$$

By (IH), $Q_1' \in S_1$ and $Q_1 \approx_{s'} Q_1'$ whenever $s \not\sqsubseteq s'$ and l is receiving at s. Hence $(\boldsymbol{\nu}\,\vec{x})(Q_1'|Q_2) \in S$ and, repeatedly using Proposition 9, $(\boldsymbol{\nu}\,\vec{x})(Q_1|Q_2) \approx_{s'} (\boldsymbol{\nu}\,\vec{x})(Q_1'|Q_2)$ whenever l is receiving at s and $s \not\sqsubseteq s'$.

The other case, communication between Q_1 and Q_2, is dealt with just as straightforwardly. ■

Proposition 10 allows us to consistently integrate the secrecy typing of [20, 23] and the present behavioural theory, for the secrecy analysis in processes and, via embedding, in programs. For example, given a $\lambda_{()\times+}$-term MN, we can check the secrecy of $[\![M]\!]_m$ by typing, $[\![N]\!]_n$ by behavioural secrecy, and finally verify their combination using typing. Another consequence of Proposition 10 is a simple proof of the following noninterference result, first given in [23]. We first start from the following lemma.

Lemma 8. *Assume that* $\vdash P_{1,2} \triangleright A$ *and* $\mathsf{tamp}(A) \not\sqsubseteq s$ *then* $P_1^A \approx_s P_2^A$.

Proof. We define

$$\mathcal{R} = \{(P^B, Q^C) \mid B \bowtie C, \mathsf{tamp}(B \restriction \mathsf{fn}(B) \cap \mathsf{fn}(C)) \not\sqsubseteq s\}.$$

Clearly $\mathcal{R}$ is semi-typed. To see that it is an s-bisimulation let $(P^B, Q^C) \in \mathcal{R}$, $P^B \xrightarrow{l} {P'}^{B'}$ where x is the subject of l. Let $\Delta = \mathsf{fn}(B) \cap \mathsf{fn}(C)$. If $x \notin \mathsf{fn}(C)$ then l is Δ-invisible and hence s-Δ-invisible. Then $Q^C \xrightarrow{\hat{l}} Q^C$ and thus $Q^C \overset{\hat{l}}{\Longrightarrow}_{\Delta,s} Q^C$. Clearly this transition is also Δ-branching closed. Easily, $B' \bowtie C$. But because the name we loose in the transition, x, is not in $\mathsf{fn}(C)$ and the new ones we import by l are fresh ${P'}^{B'}\ \mathcal{R}\ Q^C$, as required. If $x \in \mathsf{fn}(C)$ we must distinguish two cases.

- If l is an output, then it is s-Δ-invisible. The reason for this depends on whether l is linear or not. In the former case, by assumptions, $\mathsf{sec}(B(x)) \not\sqsubseteq s$ because $\mathsf{tamp}(B) \not\sqsubseteq s$. In the latter case s-Δ-invisibility is a consequence of Δ-invisibility which in turn follows from l being non-affecting. In either case then $Q^C \xrightarrow{\hat{l}} Q^C$ and we can proceed as in the case above where $x \notin \mathsf{fn}(C)$.
- If l is an input, we use input availability (Lemma 4 (3) to infer the existence of a transition $Q^C \xrightarrow{l} {Q'}^{C'}$). In this case easily $B' \bowtie C'$, i.e. ${P'}^{B'}\ \mathcal{R}\ {Q'}^{C'}$, as required. ■

Corollary 3. (noninterference) *Let* $\vdash P_{1,2} \triangleright A$ *be typable by the secrecy typing rules in Appendix E and* $\mathsf{tamp}(A) \not\sqsubseteq s$. *Then* $\vdash P_1 \cong_s P_2 \triangleright A$.

Proof. By Lemma 8 and Propositions 10 and 9. ■

We conclude this section by an example of reasoning about the secure λ-term mentioned in the introduction.

Example 7. (secrecy via encoding) Let $M \stackrel{\text{def}}{=} \mathtt{case}\, y^{\top}\, \mathtt{of}\, \{\mathtt{in}_i() : \mathtt{in}_1(\star)\}_{i\in\{1,2\}}$. We show $[\![M : \mathtt{bool}_{\top}]\!]_u \stackrel{\text{def}}{=} !u(c).\overline{y}(e)e[.\overline{c}\mathtt{in}_1 \& .\overline{c}\mathtt{in}_1] \triangleright u : (\mathbb{B}_{\bot})^{!} \to \underline{y} : (\overline{\mathbb{B}_{\top}})^{?}$ is secure. By Proposition 9, it suffices to show $e[.\overline{c}\mathtt{in}_1 \& .\overline{c}\mathtt{in}_1] \triangleright e : \overline{\mathbb{B}}_{\top} \to c : \mathbb{B}_{\bot}$ is secure. But this has already been shown in Example 6, hence done.

Remark 4. (extensions to other type structures) We have presented a theory of behavioural secrecy focussing on the pure linear π-calculus. The framework is systematically extendible to other type structures which integrate linearity with affinity (nontermination) [7], statefulness (references) [23], control [17] and nondeterminism [20]. In each case, the only necessary extensions are (1) the incorporation of a new s-affecting action into s-bisimilarity and its dual receiving action into behavioural secrecy, and (2) when affinity (nontermination) is in the type structure, we change Definition 1 as follows: $\mathbb{B}$ becomes $()^{\uparrow_A}$ ($\uparrow_A$ indicates possibly diverging, or affine, output), and the condition "$C[P_1] \Downarrow^1_u$ and $C[P_2] \Downarrow^2_u$" becomes "$C[P_i] \Downarrow_x$ and $C[P_j] \Uparrow_x$ with $i \neq j$" (here $\Downarrow_x$ iff $P \longrightarrow^* \overline{x}|P'$ for some P', and $\Uparrow_x$ iff not $\Downarrow_x$). Except for these two changes, Definitions 1–8 can be used without change. To elaborate further:

- To extend this calculus to the affine type structures of [7], we first check Definition 1, redefined as above. Clearly, any unary affine output is affecting, hence both unary and branching affine outputs are categorised as affecting in the same way as linear branching output in Table 1. They are however *not* enabling, hence affine inputs are not enabled, unlike linear inputs. At the level of secrecy, this means we abstract away, in $\approx_s$, affine outputs with high secrecy levels, just as linear selection; dually, affine inputs is taken care of in behavioural secrecy. We also note that mixing linearity and affinity in a single type structure is easily done consistently, see [23].
- To extend the present theory to non-determinism [20], we add reference agents [23] or, equivalently, recursion [20] and incorporate stateful replicated inputs/outputs which are given to (in the case of references) "write" actions. Using Definition 1 or its affine version mentioned above, we can show that stateful replicated outputs are affecting and enabled. The secrecy refinement works as in the previous example. This allows us to infer, via an embedding, that the following imperative program is behaviourally secure (assuming $\bot \lneq s \lneq \top$):

  $$\mathtt{if}\ w^{s}\mathtt{then}\ z^{\top} :=\!!x^{\bot}; y^{\bot} := 1\ \mathtt{else}\ y^{\bot} := 1$$

  The encoding of this program, given in [20, 23], allows to neglect the low-level "reading" action at x and the high-level writing at z because they

are categorised as (non-stateful) replicated output. On the other hand, the action at y which is stateful and hence affecting at $\perp$, leads to the same effect in both branches so the two branches are s-bisimilar. This warrants us to conclude that the encoding is behaviourally secure. Equational and operational correspondence allows to lift this result to the source program, so it must be secure.

Mixing linearity with other type structures (cf. [23]) can be done so that we can ensure the same liveness property for call sequences (Lemma 2 (3)) in each extension, which allows us to apply the same proof methods to obtain the corresponding results such as Theorem 1 and Proposition 9. Together with full abstraction, the framework offers a basis for uniform behavioural analysis of secrecy in programming languages.

References

1. Abadi, M., Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999.
2. Abadi, M., Secrecy in programming-language semantics, *MFPS XV*, ENTCS, 20 (April 1999).
3. Abadi, M., Banerjee, A., Heintze, N. and Riecke, J., A core calculus of dependency, *POPL'99*, ACM, 1999.
4. Abadi, M. and Gordon, D., A Bisimulation Method for Cryptographic Protocols, Nordic Journal of Computing 5, 4 (Winter 1998), 267-303.
5. Abadi, M., Fournet, C. and Gonthier, G., Authentication Primitives and their Compilation, POPL'00, 302-315, ACM, 2000.
6. Abramsky, S., Jagadeesan, R. and Malacaria, P., Full Abstraction for PCF. *Info. & Comp.* 163 (2000), 409-470.
7. Berger, M., Honda, K. and Yoshida, N., Sequentiality and the π-Calculus, *TLCA01*, LNCS 2044, pp.29-45, Springer, 2001.
8. Boudol, G. and Castellani, I., Noninterference for Concurrent Programs, *ICALP01*, LNCS, Springer, 2001.
9. Focardi, R. and Gorrieri, R., The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transactions on Software Engineering,* 23(9), 1997.
10. Focardi, R., Gorrieri, R. and Martinelli, F., Non-interference for the analysis of cryptographic protocols. *ICALP00*, LNCS 1853, Springer, 2000.
11. Gay, S. and Hole, M., Types and Subtypes for Client-Server Interactions, *ESOP'99*, LNCS 1576, 74–90, Springer, 1999.
12. Girard, J.-Y., Linear Logic, *TCS*, Vol. 50, 1–102, 1987.
13. Hennessy, M. and Riely, J., Information flow vs resource access in the asynchronous pi-calculus, *ICALP00*, LNCS 1853, 415-427, Springer, 2000.
14. Honda, K., Types for Dyadic Interaction. *CONCUR'93*, LNCS 715, 509-523, 1993.
15. Honda, K., Composing Processes, *POPL'96*, 344-357, ACM, 1996.
16. Honda, K., Elementary Structures in Process Theory (1): Sets with Renaming, *Journal of Mathematical Structures in Computer Science* (2000), Vol. 10, Cambridge University Press, October, 2000.
17. Honda, K., Notes on the linear π-calculus and LLP, June, 2001.

18. Honda, K., Kubo, M. and Vasconcelos, V., Language Primitives and Type Discipline for Structured Communication-Based Programming. *ESOP'98*, LNCS 1381, 122–138. Springer-Verlag, 1998.
19. Honda, K. and Tokoro, M. An object calculus for asynchronous communication. ECOOP'91, LNCS, 1991.
20. Honda, K., Vasconcelos, V. and Yoshida, N., Secure Information Flow as Typed Process Behaviour, *ESOP'00*, LNCS 1782, 180–199, 2000. Full version: MCS report 2000-01, University of Leicester, available at www.mcs.ac.uk/~nyoshida.
21. Honda, K. and Yoshida, N. On Reduction-Based Process Semantics. *TCS*. 151, 437-486, 1995.
22. Honda, K. and Yoshida, N., Game-theoretic analysis of call-by-value computation. *TCS*, 221 (1999), 393–456, 1999.
23. Honda, K. and Yoshida, N., A uniform type structure for secure information flow, To appear in *POPL'02*, ACM, 2002.
24. Hyland, M. and Ong, L., "On Full Abstraction for PCF": I, II and III. *Info. & Comp.* 163 (2000), 285-408.
25. Kobayashi, N., Pierce, B., and Turner, D., Linear types and π-calculus, *POPL'96*, 358–371, 1996.
26. Milner, R., Functions as Processes. *MSCS*, 2(2), 119–146, CUP, 1992.
27. Milner, R., Parrow, J.G. and Walker, D.J., A Calculus of Mobile Processes, *Info. & Comp.* 100(1), pp.1–77, 1992.
28. Philippou, A. and Walker, D., On confluence in the π-Calculus, *ICALP'97*, LNCS 1256, 314–324, Springer, 1997.
29. Sangiorgi, D., The name discipline of uniform receptiveness, *ICALP'97*, LNCS 1256, 303–313, Springer, 1997.
30. Sabelfield, A. and Sands, D. A per model of secure information flow in sequential programs. *ESOP'99*, LNCS 1576, Springer, 1999.
31. Smith, G., A New Type System for Secure Information Flow, *CSFW'01*, IEEE, 2001.
32. Smith, G. and Volpano, D., *Secure information flow in a multi-threaded imperative language*, pp.355–364, *POPL'98*, ACM, 1998.
33. Vasconcelos, V., Typed concurrent objects. *ECOOP'94*, LNCS 821, pp.100–117. Springer, 1994.
34. Yoshida, N. Graph Types for Mobile Processes. *FST/TCS'16*, LNCS 1180, pp.371–386, Springer, 1996. The full version as LFCS Technical Report, ECS-LFCS-96-350, 1996.
35. Yoshida, N., Berger, M. and Honda, K., Strong Normalisation in the π-Calculus, *LICS'01*, IEEE, 2001. The full version as MCS technical report, 2001-09, University of Leicester, 2001. Available at www.mcs.le.ac.uk/~nyoshida.

A Structural Equality

The relation $\equiv$ is the least congruence generated by $\equiv_\alpha$ and the following equations.

$$
\begin{array}{llll}
P|\mathbf{0} \equiv P & P|Q \equiv Q|P & (P|Q)|R \equiv P|(Q|R) & \\
(\boldsymbol{\nu}\, x)\mathbf{0} \equiv \mathbf{0} & (\boldsymbol{\nu}\, xy)P \equiv (\boldsymbol{\nu}\, yx)P & ((\boldsymbol{\nu}\, x)P)|Q \equiv (\boldsymbol{\nu}\, x)(P|Q) & (x \notin \mathsf{fn}(Q)) \\
(\overline{x}\mathtt{in}_i(\vec{y})P)|Q \equiv \overline{x}\mathtt{in}_i(\vec{y})(P|Q) & & \text{if } \{\vec{y}\} \cap \mathsf{fn}(Q) = \emptyset & \\
(\boldsymbol{\nu}\, z)\overline{x}\mathtt{in}_i(\vec{y})P \equiv \overline{x}\mathtt{in}_i(\vec{y})(\boldsymbol{\nu}\, z)P & & \text{if } z \notin \{x\vec{y}\} & \\
\overline{x}\mathtt{in}_i(\vec{y})\overline{z}\mathtt{in}_j(\vec{w})P \equiv \overline{z}\mathtt{in}_j(\vec{w})\overline{x}\mathtt{in}_i(\vec{y})P & & \text{if } z \notin \{\vec{y}\} \text{ and } x \notin \{\vec{w}\} &
\end{array}
$$

We omit rules for unary prefixes since they are subsumed by those for branching/selection.

B Action Types

We first reiterate the definition of action types. An *action type*, denoted $A, B, \ldots$, is a finite directed graph with nodes of the form $x : \tau$, such that:

- no names occur twice; and
- edges are of the form $x : \tau \to y : \tau'$ such that either (1) $\mathsf{md}(\tau) = \downarrow$ and $\mathsf{md}(\tau') = \uparrow$ or (2) $\mathsf{md}(\tau) = \mathbf{!}$ and $\mathsf{md}(\tau') = \mathbf{?}$.

We write $x \to y$ if $x : \tau \to y : \tau'$ for some τ and τ', in a given action type. If x occurs in A and for no y we have $y \to x$ then we say x is *active in* A. $|A|$ (resp. $\mathsf{fn}(A)$, $\mathsf{sbj}(A)$, $\mathsf{md}(A)$) denotes the set of nodes (resp. names, active names, modes) in A. We often write $x : \tau \in A$ instead of $x : \tau \in |A|$, and write $A(x)$ for the channel type assigned to x in A. $A \backslash \vec{x}$ is the result of taking off nodes with names in $\vec{x}$ from A. $A \otimes B$ is the graph union of A and B, with the condition that $\mathsf{fn}(A) \cap \mathsf{fn}(B) = \emptyset$.

The symmetric partial operator $\odot$ on types is already given in Section 2. We then write $A \asymp B$ iff:

- whenever $x : \tau \in A$ and $x : \tau' \in B$, $\tau \odot \tau'$ is defined; and
- whenever $x_1 \to x_2$, $x_2 \to x_3$, $\ldots$, $x_n \to x_{n+1}$ (alternately in A and B), we have $x_1 \neq x_{n+1}$.

Then $A \odot B$, defined iff $A \asymp B$, is the following action type.

- $x : \tau \in |A \odot B|$ iff either (1) $x \in (\mathsf{fn}(A) \backslash \mathsf{fn}(B)) \cup (\mathsf{fn}(B) \backslash \mathsf{fn}(A))$ and $x : \tau$ occurs in A or B; or (2) $x : \tau' \in A$ and $x : \tau'' \in B$ and $\tau = \tau' \odot \tau''$.
- $x \to y$ in $A \odot B$ iff $x = z_1 \to z_2$, $z_2 \to z_3, \ldots, z_{n-1} \to z_n = y$ alternately in A and B ($n \geq 2$) and, moreover, for no w we have $w \to x$ and for no w' we have $y \to w'$ in A or B.

We can easily check that $\odot$ is a symmetric and associative partial operation on action types with unit $\emptyset$.

C Typing Rules

$A\langle \vec{y} : \vec{\tau} \rangle$ indicates each $y_i : \tau_i$ occurs in A. $x : \tau \to A$ adds new edges from $x : \tau$ to the non-suppressed nodes in A. A^{-x} indicates $x \notin \mathsf{fn}(A)$. We assume $\uparrow A$ in ($\mathsf{In}^{\downarrow}$) and ($\mathsf{Bra}^{\downarrow}$) is either a singleton or empty ("unique-answer-per-thread").

$$\frac{}{\vdash \mathbf{0} \triangleright _} \ (\mathsf{Zero}) \qquad \frac{\vdash P_i \triangleright A_i \quad (i = 1, 2) \qquad A_1 \asymp A_2}{\vdash P_1 | P_2 \triangleright A_1 \odot A_2} \ (\mathsf{Par})$$

$$\frac{\vdash P \triangleright A\langle x : \tau \rangle \qquad \mathsf{md}(\tau) \in \{*, \mathbf{!}\}}{\vdash (\boldsymbol{\nu}\, x) P \triangleright A/x} \ (\mathsf{Res}) \qquad \frac{\vdash P \triangleright A^{-x} \qquad \mathsf{md}(\tau) \in \{*, \mathbf{?}\}}{\vdash P \triangleright A \otimes x : \tau} \ (\mathsf{Weak}\text{-}*, \mathbf{?})$$

$$\frac{\vdash P \triangleright \vec{y} : \vec{\tau} \otimes \uparrow A^{-x} \otimes ?B^{-x}}{\vdash x(\vec{y}).P \triangleright (x : (\vec{\tau})^{\downarrow} \to A) \otimes B}\ (\mathsf{In}^{\downarrow}) \qquad \frac{\vdash P \triangleright \vec{y} : \vec{\tau} \otimes ?A^{-x}}{\vdash !\, x(\vec{y}).P \triangleright x : (\vec{\tau})^{!} \to A}\ (\mathsf{In}^{!}) \qquad \frac{\vdash P \triangleright C\langle \vec{y} : \vec{\tau} \rangle \quad C/\vec{y} = A \asymp x : (\vec{\tau})^{p}}{\vdash \overline{x}(\vec{y})P \triangleright A \odot x : (\vec{\tau})^{p}}\ (\mathsf{Out})\ (p \in \{\uparrow, ?\})$$

$$\frac{\vdash P_i \triangleright \vec{y}_i : \vec{\tau}_i \otimes \uparrow A^{-x} \otimes ?B^{-x}}{\vdash x[\&_i(\vec{y}_i).P_i] \triangleright (x : [\&_i \vec{\tau}_i]^{\downarrow} \to A) \otimes B}\ (\mathsf{Bra}^{\downarrow}) \qquad \frac{\vdash P_i \triangleright \vec{y}_i : \vec{\tau}_i \otimes ?A^{-x}}{\vdash !\, x[\&(\vec{y}_i).P_i] \triangleright x : [\&_i \vec{\tau}_i]^{!} \to A}\ (\mathsf{Bra}^{!}) \qquad \frac{\vdash P \triangleright C\langle \vec{y} : \vec{\tau}_j \rangle \quad C/\vec{y} = A \asymp x : [\oplus_i \vec{\tau}_i]^{p}}{\vdash \overline{x}\mathtt{in}_j(\vec{y})P \triangleright A \odot x : [\oplus_i \vec{\tau}_i]^{p}}\ (\mathsf{Sel})\ (p \in \{\uparrow, ?\})$$

D Transition Rules

We assume all l.h.s. processes are well-typed. A allows l unless: (1) $A(\mathsf{sbj}(l)) = *$ or (2) l is output and $\mathsf{md}(A(\mathsf{sbj}(l))) = \mathbf{!}$, cf. [7]. $\mathsf{n}(l)$ is the set of names in l.

$$\begin{array}{ll}
x[\&_i \vec{y}_i.P_i]^{\,A} \xrightarrow{x\mathtt{in}_i(\vec{y}_i)} P_i^{\,\vec{y}_i:\vec{\tau}_i \otimes A/x} & (x : [\&\vec{\tau}_i]^{\downarrow} \in A) \\
!x[\&_i \vec{y}_i.P_i]^{A} \xrightarrow{x\mathtt{in}_i(\vec{y}_i)} !x[\&_i \vec{y}_i.P_i] | P_i^{\,\vec{y}_i:\vec{\tau}_i \otimes A} & (x : [\&\vec{\tau}_i]^{!} \in A) \\
\overline{x}\mathtt{in}_i(\vec{y})P^{\,A} \xrightarrow{\overline{x}\mathtt{in}_i(\vec{y})} P^{\,\vec{y}:\vec{\tau}_i \otimes A/x} & (x : [\oplus_i \vec{\tau}_i]^{\uparrow} \in A) \\
\overline{x}\mathtt{in}_i(\vec{y}).P^{\,A} \xrightarrow{\overline{x}\mathtt{in}_i(\vec{y})} P^{\,\vec{y}:\vec{\tau}_i \otimes A} & (x : [\oplus_i \vec{\tau}_i]^{?} \in A)
\end{array}$$

$$\frac{P_1' \equiv_\alpha P_1 \;\; P_1^{A_1} \xrightarrow{l} P_2^{A_2} \;\; P_2 \equiv_\alpha P_2'}{P_1'^{\,A_1} \xrightarrow{l} P_2'^{\,A_2}} \qquad \frac{P_1^{A_1} \xrightarrow{l} P_2^{A_2} \quad x \notin \mathsf{n}(l)}{(\boldsymbol{\nu}\, x)P_1^{\,A_1/x} \xrightarrow{l} (\boldsymbol{\nu}\, x)P_2^{\,A_2/x}}$$

$$\frac{P_1^{A_1} \xrightarrow{l} P_2^{A_2} \quad A_1 \odot B \text{ allows } l}{P_1|Q^{\,A_1 \odot B} \xrightarrow{l} P_2|Q^{\,A_2 \odot B}} \qquad \frac{P_1^{A_1} \xrightarrow{l} P_2^{A_2} \quad Q_1^{B_1} \xrightarrow{\overline{l}} Q_2^{B_2}}{P_1|Q_1^{\,A_1 \odot B_1} \xrightarrow{\tau} (\boldsymbol{\nu}\, \mathsf{bn}(l))(P_2|Q_2)^{\,A_2 \odot B_2/\mathsf{bn}(l)}}$$

$$\frac{P_1^{A_1} \xrightarrow{l} P_2^{A_2} \quad \mathsf{n}(l) \cap \{\vec{y}\} = \emptyset}{\overline{x}\mathtt{in}_i(\vec{y})P_1^{\,A_1/\vec{y} \odot x:[\oplus_i \vec{\tau}_i]^p} \xrightarrow{l} \overline{x}\mathtt{in}_i(\vec{y})P_2^{\,A_2/\vec{y} \odot x:[\oplus_i \vec{\tau}_i]^p_i}}$$

$$\frac{P_1^{A_1} \xrightarrow{x\mathtt{in}_i(\vec{z})} P_2^{A_2}}{\overline{x}\mathtt{in}_i(\vec{y})P_1^{\,A_1/\vec{y} \odot x:[\oplus_i \vec{\tau}_i]^p} \xrightarrow{\tau} (\boldsymbol{\nu}\, \vec{y})P_2\{\vec{y}/\vec{z}\}^{\,A_2/\vec{z}}}$$

We omit rules for unary actions and symmetric case of $|$. The rules are well-typed in the sense that if $P_1^{A_1}$ is well-typed and $P_1^{A_1} \xrightarrow{l} P_2^{A_2}$ then $P_2^{A_2}$ is well-typed.

E Secrecy Typing

We first give the typing rules in which (1) annotated channel types are used, and (2) no secrecy is incorporated into the typing rules. Thus the set of typable terms are precisely the same as those given by Appendix C, and the secrecy properties (e.g. noninterference) are not ensured. The notations are as before.

$$\frac{}{\vdash \mathbf{0} \triangleright _}\ (\mathsf{Zero}) \qquad \frac{\vdash P_i \triangleright A_i \quad (i=1,2) \quad A_1 \asymp A_2}{\vdash P_1|P_2 \triangleright A_1 \odot A_2}\ (\mathsf{Par}) \qquad \frac{\vdash P \triangleright A\langle x:\tau\rangle \quad \mathsf{md}(\tau) \in \{*, \mathbf{!}\}}{\vdash (\boldsymbol{\nu}\, x)P \triangleright A/x}\ (\mathsf{Res}) \qquad \frac{\vdash P \triangleright A^{\text{-}x} \quad \mathsf{md}(\tau) \in \{*, \mathbf{?}\}}{\vdash P \triangleright A \otimes x:\tau}\ (\mathsf{Weak}\text{-}*, \mathbf{?})$$

$$\frac{\vdash P \triangleright \vec{y}:\vec{\tau} \otimes \uparrow A^{\text{-}x} \otimes \mathbf{?} B^{\text{-}x}}{\vdash x(\vec{y}).P \triangleright (x:(\vec{\tau})^{\downarrow}_{s} \rightarrow A) \otimes B}\ (\mathsf{In}^{\downarrow}) \qquad \frac{\vdash P \triangleright \vec{y}:\vec{\tau} \otimes \mathbf{?} A^{\text{-}x}}{\vdash \mathbf{!}\, x(\vec{y}).P \triangleright x:(\vec{\tau})^{\mathbf{!}}_{s} \rightarrow A}\ (\mathsf{In}^{\mathbf{!}}) \qquad \frac{\vdash P \triangleright C\langle \vec{y}:\vec{\tau}\rangle \quad C/\vec{y} = A \asymp x:(\vec{\tau})^{p}_{s}}{\vdash \overline{x}(\vec{y})P \triangleright A \odot x:(\vec{\tau})^{p}_{s}}\ (\mathsf{Out})\ (p \in \{\uparrow, \mathbf{?}\})$$

$$\frac{\vdash P_i \triangleright \vec{y}_i:\vec{\tau}_i \otimes \uparrow A^{\text{-}x} \otimes \mathbf{?} B^{\text{-}x}}{\vdash x[\&_i(\vec{y}_i).P_i] \triangleright (x:[\&_i\vec{\tau}_i]^{\downarrow}_{s} \rightarrow A) \otimes B}\ (\mathsf{Bra}^{\downarrow}) \qquad \frac{\vdash P_i \triangleright \vec{y}_i:\vec{\tau}_i \otimes \mathbf{?} A^{\text{-}x}}{\vdash \mathbf{!}\, x[\&(\vec{y}_i).P_i] \triangleright x:[\&_i\vec{\tau}_i]^{\mathbf{!}}_{s} \rightarrow A}\ (\mathsf{Bra}^{\mathbf{!}}) \qquad \frac{\vdash P \triangleright C\langle \vec{y}:\vec{\tau}_j\rangle \quad C/\vec{y} = A \asymp x:[\oplus_i\vec{\tau}_i]^{p}_{s}}{\vdash \overline{x}\mathtt{in}_j(\vec{y})P \triangleright A \odot x:[\oplus_i\vec{\tau}_i]^{p}_{s}}\ (\mathsf{Sel})\ (p \in \{\uparrow, \mathbf{?}\})$$

In the linear type structure, the secrecy of processes is ensured simply by replacing the rule ($\mathsf{Bra}^{\downarrow}$) above by the following one. Other rules remain the same.

$$\frac{\vdash P_i \triangleright \vec{y}_i:\vec{\tau}_i \otimes \uparrow A^{\text{-}x} \otimes \mathbf{?} B^{\text{-}x} \quad s \sqsubseteq \mathsf{tamp}(A)}{\vdash x[\&_i(\vec{y}_i).P_i] \triangleright (x:[\&_i\vec{\tau}_i]^{\downarrow}_{s} \rightarrow A) \otimes B}\ (\mathsf{Bra}^{\downarrow})$$

We call the resulting set of processes, *securely typed processes.*

F Proofs in Section 4

F.1 Proofs for Lemma 3

In the following, we often denote $l \equiv_{\alpha} l'$ if they originates from the same index and $\equiv$ denotes the structure *equivalence* (hence $P \equiv Q \ \Rightarrow\ P \approx_{\mathrm{L}} Q$).

(1) Standard.

(2) Let $\Delta = \mathsf{fn}(A)$. Assume $\mathcal{R} = \{((P|R_1),\ (P|R_2)) \mid P \text{ and } R_i \text{ as in Lemma 3 (2)}\}$. We prove $\mathcal{R}$ is a bisimulation using Lemma 3 (1).

The case $P \mid R_1 \stackrel{l}{\longrightarrow} P' \mid R_1$ with $P \stackrel{l}{\longrightarrow} P'$ is trivial.

The case $P \mid R_1 \stackrel{l}{\longrightarrow} P \mid R_1'$ with $R_1 \stackrel{l}{\longrightarrow} R_1'$. First, suppose $l = x(\vec{y})$ with $\mathsf{md}(l^A) = \uparrow$. Then we have $\vdash R_1' \triangleright A_1 \otimes B_1/x \otimes \vec{y}:\vec{\tau}$. Obviously $\mathsf{sbj}(l) \notin \Delta$, so l is invisible. Next, since $\mathsf{fn}(P) \cap \{\vec{y}\} = \emptyset$, P and R_1' only share replicated output names. Set $B_1' = B_1/x \otimes \vec{y}:\vec{\tau}$. Then we can check $\mathsf{fn}(B_1') \cap \mathsf{fn}(B_2) = \emptyset$. Now by $P \mid R_1 \stackrel{\varepsilon}{\Longrightarrow}_{\Delta} (P \mid R_2)$, we have $(P \mid R_2)\ \mathcal{R}\ (P \mid R_1')$, as desired. As such, we can easily check if $R_1 \stackrel{l}{\longrightarrow} R_1'$, l is invisible, and moreover it satisfies the side

condition.

The case $P \mid R_1 \xrightarrow{\boldsymbol{\tau}} (\boldsymbol{\nu}\,\vec{x})(P' \mid R_1')$ with $P \xrightarrow{l} P'$ and $R_1 \xrightarrow{\overline{l}} R_1'$. Then by definition, $\mathsf{md}(l) = \mathbf{!}$ and $\mathsf{md}(\overline{l}) = \mathbf{?}$. Let us assume $l = a(\vec{x})$. Then we can write down $P \equiv (\boldsymbol{\nu}\,\vec{w})(!a(\vec{x}).P_0 \mid Q)$ and $R_1 \equiv \overline{a}(\vec{x})R_{01}$. Hence $P \mid R_1 \xrightarrow{\boldsymbol{\tau}} \equiv (\boldsymbol{\nu}\,\vec{w})(!a(\vec{x}).P_0 \mid Q \mid (\boldsymbol{\nu}\,\vec{x})(P_0 \mid R_{01}))$. Note that $(\boldsymbol{\nu}\,\vec{x})(P_0 \mid R_{01})$ only share $\mathbf{?}$-names with $!a(\vec{x}).P_0 \mid Q$. By appropriate renaming on R_2, we have $P \mid R_1 \stackrel{\varepsilon}{\Longrightarrow}_{\Delta} (P \mid R_2) \equiv (\boldsymbol{\nu}\,\vec{w})(!a(\vec{x}).P_0 \mid Q \mid R_2')$ and $(!a(\vec{x}).P_0 \mid Q \mid R_2') \;\mathcal{R}\; (!a(\vec{x}).P_0 \mid Q \mid (\boldsymbol{\nu}\,\vec{x})(P_0 \mid R_{01}))$. Hence by applying (1), we conclude the proof.[6]

(3,4) Similar with Proposition 3 (1).

(5) Case $\boldsymbol{\tau}$: We show $\mathcal{R} = \{(P,\ P') \mid P \xrightarrow{\boldsymbol{\tau}} P'\} \cup \equiv_{\alpha}$ is a bisimulation.

Suppose $P \xrightarrow{l} R$. Then if $l \neq \boldsymbol{\tau}$ or indices of two tau actions are distinct, then by Proposition 4 (1,2), we know $P \xrightarrow{l} R'$ with $R \xrightarrow{\boldsymbol{\tau}} R'\,\mathcal{R}\,R$, as required. If $l = \boldsymbol{\tau}$ and indices of two tau actions are the same, then $R' \equiv_{\alpha} P'$, hence done.

Next suppose $P' \xrightarrow{l} R$. Since $\boldsymbol{\tau}$ is strongly invisible, $P \stackrel{l}{\Longrightarrow} R \equiv_{\alpha} R$, as required.

Case $\mathsf{md}(l) = \mathbf{!}$: Suppose $P \xrightarrow{l} P'$. Then we can easily check $(P,\ P')$ is in the relation defined in (2), by assigning $R_1 \equiv_{\alpha} \mathbf{0}$. Hence done.

Case $\mathsf{md}(l) = \uparrow$: We show $\mathcal{R} \stackrel{\text{def}}{=} \{(P,\ P') \mid P \xrightarrow{\overline{a}(\vec{x})} P'\} \cup \approx_{\text{L}}$ is a bisimulation. There are two interesting cases. Other cases are just similar with the case for $\boldsymbol{\tau}$. Let $\Gamma = \mathsf{fn}(P) \cap \mathsf{fn}(P')$.

Suppose $P \xrightarrow{\overline{a}(\vec{x}')} P'\{x_i'/x_i\}$ where two actions are occurred at the same indices, i.e. P gets different bound names x_i and x_i' from the outside. Then we can write down $P' \equiv (\boldsymbol{\nu}\,\vec{w})(\prod_i Q_i \mid R)$ and $P'\{x_i'/x_i\} \equiv (\boldsymbol{\nu}\,\vec{w})(\prod_i Q_i\{x_i'/x_i\} \mid R)$ with $R_i \equiv !x_i(\vec{w}_i).Q_i'$ or $!x_i[\&(\vec{w}_{ij}).Q_{ij}']$. Then we can apply Lemma 3 (2), defining $P \stackrel{\text{def}}{=} R$, $R_1 \stackrel{\text{def}}{=} \prod_i Q_i$ and $R_2 \stackrel{\text{def}}{=} \prod_i Q_i\{x_i'/x_i\}$. Hence $P' \approx_{\text{L}} P'\{x_i'/x_i\}$. Since a is a linear output name, $a \notin \mathsf{fn}(P')$, so $a \notin \Gamma$. Thus $\overline{a}(\vec{x})$ is a Γ invisible. Hence by $P' \stackrel{\varepsilon}{\Longrightarrow} P'$, we have $P'\{x_i'/x_i\}\mathcal{R}P'$, as desired.

Suppose $P' \xrightarrow{l} R$ for some l. If $\overline{a}(\vec{x}) \not\frown l$, then we finish the proof by confluence. So assume $\overline{a}(\vec{x}) \frown l$. Then obviously $P \xrightarrow{\overline{a}(\vec{x})} P' \xrightarrow{l} R$. We need to show $P \stackrel{l}{\Longrightarrow}_{\Gamma} R$. Since a is a linear output name, $a \notin \mathsf{fn}(P')$, so $a \notin \Gamma$. Thus $\overline{a}(\vec{x})$ is a Γ-strong invisible. Moreover $P \stackrel{l}{\Longrightarrow}_{\Gamma} R$ is a Γ-branching closed. Hence by $R\mathcal{R}R$, we have closed this case. The case $\overline{a}\mathtt{in}_i(\vec{x})$ is just similar.

(5) We show $\mathcal{R} \stackrel{\text{def}}{=} \{(P,\ P') \mid P \xrightarrow{\overline{a}(\vec{x})} P'\} \cup \approx_{\text{L}}$ is a bisimulation. Just similar with the case of $\uparrow$ above by using confluence, except the following case, which

[6] While we can use (5) for the case of $\boldsymbol{\tau}$, the given proof has an advantage in that it is adaptable to the setting where we extend the calculus to the one with non-determinism and state.

corresponds to the former case of $\uparrow$ above. Suppose $P \xrightarrow{a(\vec{x}')} P'\{x_i'/x_i\}$ where two actions $a(\vec{x})$ and $a(\vec{x}')$ are the same indices, i.e. P gets different bound names x_i and x_i' from the outside. Then this time, we can write down $P' \equiv (\boldsymbol{\nu}\, \vec{w})(\prod_i Q_i | R)$ and $P'\{x_i'/x_i\} \equiv (\boldsymbol{\nu}\, \vec{w})(\prod_i Q_i\{x_i'/x_i\} \mid R)$ with $R_i \equiv \overline{x}_i(\vec{w}_i).Q_i'$ or $\overline{x}_i\mathtt{in}_j(\vec{w}_j).Q_j'$. This time we can apply Lemma 3 (3), hence we have $P' \approx_{\mathrm{L}} P'\{x_i'/x_i\}$. The rest is the same as the former case of $\uparrow$ above.

Remark 5. The above lemma does not hold for the branching input. Consider the case $P \xrightarrow{a\mathtt{in}_i(\vec{x})} P'$ and $P' \xrightarrow{l} R$ for some l such that $\overline{a}(\vec{x}) \not\frown l$. Then by the same reasoning as the linear output case, we have $P \xRightarrow{l}_\Gamma R$ since the action $a\mathtt{in}_i(\vec{x})$ is invisible. However this transition is not always branching closed, since we may have $P \xrightarrow{a\mathtt{in}_j(\vec{x})} P_j' \xrightarrow{l_j} P_j''$ with $l_j \neq l$ (i.e. l_j is a selection valiant of l).

F.2 Proofs for Lemma 4

(1): Suppose $\vdash P \triangleright A$. Assume $P \approx_{\mathrm{L}}^{\Delta} Q$ and $P \xrightarrow{l} P'$ where l satisfies either (1,2,3) in (1). Then by Lemma 3 (5) and (6), we always have $P \approx_{\mathrm{L}} P'$. Note, for each case, we can easily verify that $\Delta \subseteq \mathsf{fn}(P) \cap \mathsf{fn}(P')$ and $\Delta = \mathsf{fn}(P') \cap \mathsf{fn}(Q)$, as similar with the proofs in Lemma 3 (5) and (6). Hence by the transitivity, we have $Q \approx_{\mathrm{L}} P'$, as required.

(2): We show $\mathcal{R} \stackrel{\text{def}}{=} \{(P,\ Q) \mid \text{as in Lemma 4 (2)}\} \cup \approx_{\mathrm{L}}$ is a bisimulation.

Case 1: Assume $P \xrightarrow{l}_\Delta P'$ with l not Δ-s.i. First, by confluence, $P_j \xrightarrow{l} P_j'$. Then by assumption, $P_j \approx_{\mathrm{L}}^{\Delta} Q$, there exists a Δ-b.c. such that $\{Q \xRightarrow{\hat{l}}_\Delta Q_k\}$ with $Q_k \approx_{\mathrm{L}}^{\Delta'} P_j'$. We can easily check $\Delta' \stackrel{\text{def}}{=} \mathsf{fn}(P') \cap \mathsf{fn}(Q_k) = \mathsf{fn}(P_j') \cap \mathsf{fn}(Q_k)$ for each j. Hence $P'\mathcal{R}Q_k$ for each k, as required.

Case 2: Assume $P \Longrightarrow_\Delta P_j \stackrel{\text{def}}{\Leftrightarrow} P \xrightarrow{l_{1j}\ldots l_{nj}} P_j$ and $P \xrightarrow{l}_\Delta P'$ with $l \equiv_\alpha l_{ij}$ for some $0 \leq i \leq n$. Then by (permutation lemma, Lemma 2 (1)), $P \xrightarrow{l_{ij}} \Longrightarrow_\Delta P_j$. Hence by definition of the transition relation, there exists an injective renaming σ such that $P \xrightarrow{l} P' \Longrightarrow_\Delta P_j\sigma$ with (a) $\mathsf{dom}(\sigma) \cap \Delta = \emptyset$, and (b) $\mathsf{cod}(\sigma) \cap (\mathsf{fn}(P) \cup \mathsf{fn}(Q)) = \emptyset$. We also note that $\{P \Longrightarrow_\Delta P_j\sigma\}$ is a Δ-b.c. and $P_k\sigma \approx_{\mathrm{L}} Q$ for each k by Lemma 3 (4). Note that $\Delta = \mathsf{fn}(Q) \cap \mathsf{fn}(P_j\sigma)$. Hence with $Q \stackrel{\varepsilon}{\Longrightarrow} Q$, we have $P'\mathcal{R}Q$, as required.

Case 3: Suppose $Q \xrightarrow{l}_\Delta Q'$. Then by assumption, for all j, we have $\{P_j \xRightarrow{\hat{l}}_\Delta P_{ji}\}$ with $P_{ji} \approx_{\mathrm{L}}^{\Delta'} Q'$. Then by Lemma 1 (1), $\{P \Longrightarrow_\Delta P_j \xRightarrow{\hat{l}}_\Delta P_{ji}\}$ is again Δ-b.c. Hence by $P_{ij} \xrightarrow{\varepsilon} P_{ji}$, we have $Q'\mathcal{R}P_{ji}$, since the side condition of names is vacuously satisfied (by $\Delta' = \mathsf{fn}(P_{ji}) \cap \mathsf{fn}(Q')$).

(3) Suppose $P_1 \xrightarrow{l} P_2$ with $\mathsf{sbj}(l) \notin \Delta$ and l linear branching input. Then there is a Δ-b.c. $\{P_2^{A_2} \Longrightarrow_\Delta Q_{2i}^{B_{2i}}\}_{i\in I}$ such that $Q_1^{B_1} \approx_{\mathrm{L}}^{\circ} Q_{2i}^{B_{2i}}$. Then by Lemma 4 (2) above, we have $Q_1^{A_1} \approx_{\mathrm{L}}^{\circ} P_2^{A_2}$, as desired.

(4) Just similar as (3).

(5) By (1, 3, 4).

(6,7): Similarly with (2) using Lemma 2 (4) and Lemma 3 (4).

F.3 Proofs for Lemma 7

First we can easily check the side condition, "the linear output outside of Δ is not suppressed by the linear input at Δ", is always maintained as proved in the end of the proof of Proposition 8. Hence we only have to check Δ-b.c. and bisimulation. We assume P, Q_1 and Q_2 in the following is the same as in (7).

Case A: Let $\{P \stackrel{l_{1k}}{\Longrightarrow} P_k\}_{k\in K}$. Then $\mathsf{md}(l_{1k}) = \uparrow$. Let $\mathsf{sbj}(l_{k1}) = e$.

Case 1: Suppose $\mathsf{sbj}(l_{1k}) \notin \mathsf{fn}(Q_1)$. Then obviously $P \mid Q_1 \Longrightarrow_\Delta P_k \mid Q_1$, as desired.

Case 2: Suppose $\mathsf{sbj}(l_{1k}) \in \mathsf{fn}(Q_1)$. Then we have either $B_1 = e : \tau \otimes B'$ or $B_1 = c : \tau' \to e : \tau \otimes B'$ for some $c \notin \Delta$. For both cases (since $c \notin \Delta$ in the latter case), by Lemma 2 (3,4), we have a Δ-branching closed sequences $\{Q_1 \Longrightarrow_\Delta Q_{1i}\}_{i\in I}$ where

(A) for all $i \in I$, $Q_{1i} \xrightarrow{l_i} Q'_{1i} \approx_{\mathrm{L}}^{\Delta} Q_2$ by Lemma 4 (6), and
(B) for all $n \in I$, there exists $k \in K$ such that $l_n = \overline{l_{k1}}$ because $\{P \xrightarrow{l_{k1}} P_k\}$ is a branching closed.

Let us assume $Q_1 \Longrightarrow_\Delta Q_{1i} \stackrel{\text{def}}{\Leftrightarrow} Q_1 \stackrel{l_{1i}\cdots l_{ni}}{\longrightarrow}_\Delta Q_{1i}$ and $\mathsf{sbj}(l_j) \notin \Delta$. Note that, if so, for all $j \in I$, any names in transitions in $Q_1 \Longrightarrow_\Delta Q_{1j}$ are not overlapped with Δ. Then we have:

(a) $\{(P \mid Q_1) \Longrightarrow_\Delta (P \mid Q_{1i})\}_{i\in I}$ is Δ-b.c. by (A) above.
(b) for all $i \in I$, there exists $k \in K$ such that $(P \mid Q_{1i}) \xrightarrow{\tau} (P_k \mid Q'_{1i})$ by (B) above.
(c) for each $i \in I$, $\{(P \mid Q_{1i}) \xrightarrow{\tau} (P_k \mid Q'_{1i})\}$ is is Δ-b.c.

Hence by (a) and (c) above, together with Lemma 1 (1), $\{(P|Q_1) \Longrightarrow_\Delta (P_k|Q'_{1i})\}_{i\in I}$ is Δ-b.c. Then by Lemma 4 (6), we have $Q'_{1i} \approx_{\mathrm{L}} Q_1$ as required.

Case B: Let $\{P \stackrel{l_2}{\Longrightarrow} P' \xrightarrow{l_{1k}} P_k\}$ with $\mathsf{md}(l_2) = \mathbf{?}$ and $\mathsf{md}(l_{1k}) = \uparrow$ with $l_2 \frown_{\mathsf{b}} l_{1k}$. The case $\mathsf{sbj}(l_2) \notin \mathsf{fn}(Q_1)$ is the same as above. So suppose $\mathsf{sbj}(l_2) \in \mathsf{fn}(Q_1)$. Then by (input availability), we always have $Q_1 \xrightarrow{\overline{l_2}} Q'_1$ with $\mathsf{md}(l_2) = \mathbf{!}$. By Lemma 2 (5), we know $Q_1 \approx_{\mathrm{L}} Q'_1$. Note that $\{P \mid Q_1 \xrightarrow{\tau} (\boldsymbol{\nu}\, \vec{w})(P' \mid Q'_1)\}$ is Δ-b.c. Set $P = Q'_1$ and $Q_1 = P'$. Then by applying the same reasoning as above starting from $Q'_1 \mid P'$, we can get a desirable Δ-b.c., again using Lemma 1 (1).

We can easily observe that Cases A and B can be extended to $\{P \stackrel{l_1\cdots l_{1k}}{\Longrightarrow} P_k\}$ if all subjects in call-sequences from Q_1 are not overlapped with $\mathsf{fn}(P)$ by Lemma

1 (1).

Case C: Next suppose subjects in a Δ-branching closed sequences $\{Q_1 \Longrightarrow_\Delta Q_{1i}\}_{i\in I}$ in Case (A-2) overlapped with $\mathsf{fn}(P)$. If replicated output names are overlapped, then we can use Lemma 2 (5) as we proved in Case B. Suppose a linear input name in $\{Q_1 \Longrightarrow_\Delta Q_{1i}\}_{i\in I}$ is overlapped with $\mathsf{fn}(P)$. I.e. $Q_1 \stackrel{l_{i1}\cdots l_{in_i}}{\Longrightarrow} Q_{1i}$ with l_{i1} linear input with $\mathsf{sbj}(l_{i1}) \in \mathsf{fn}(P)$. Then by Lemma 2 (3, extended liveness), we have $P \Longrightarrow_\Delta P_m \xrightarrow{\overline{l_{i1}}}$. Then we can repeat the same argument as Case A. Note by (extended liveness), this routine (call sequences from $P \mid Q_1$) is always finite, so we finish the proof. ■